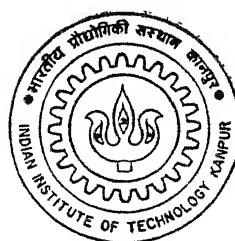


MODERN ARTIFICIAL INTELLIGENCE METHODS IN IRON AND STEEL MAKING

by

BASANT KUMAR KUKREJA



DEPARTMENT OF MATERIALS AND METALLURGICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

APRIL, 1995

MME
1995
M
KUK
MOD

TH
MME/1995/M
K 958 m

MODERN ARTIFICIAL INTELLIGENCE METHODS IN IRON AND STEEL MAKING

*A Thesis Submitted
in partial fulfilment of the Requirements
for the Degree of*

MASTER OF TECHNOLOGY

By

BASANT KUMAR KUKREJA

To the
**DEPARTMENT OF MATERIALS & METALLURGICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
April 1995**

26 JUN 1996
26 JUN 1996
CENTRAL LIBRARY
I.I.T. KANPUR

Acc. No. A. 121698

MME-1995-M-KUK-MOD



A121698

C E R T I F I C A T E

It is certified that the work contained in the thesis entitled "**Modern Artificial Intelligence Methods in Iron and Steel making**" by **BASANT KUMAR KUKREJA**, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.


Dr. Brahma Deo 24/4/95


Professor

Materials and Metallurgical

Engineering

Indian Institute of Technology

Kanpur - 208 016


Dr. Prem Kumar Kalra

Associate Professor

Electrical

Engineering

Indian Institute of Technology

Kanpur - 208 016

ACKNOWLEDGEMENT

The work in this thesis has been carried out under the supervision of Dr. P.K.Kalra and Dr. B.Deo. I am grateful to them for the support, they have extended to me during the entire duration of the thesis work.

I am grateful to Mr. Adhikari Amitabh Prasad of Nuclear Engineering Technology for his continuous advice on the finer details of the software knowledge.

Finally, I would like to thank my parents and friends, especially **Mahesh Kumar Sharma, Lalit Narain Mishra** for their moral support during my hard times in pursuing the thesis.

24 April 1995

Basant kumar Kukreja

I.I.T. - Kanpur

SYNOPSIS

Modern methods of artificial intelligence studied in this work include neural networks(NN), genetic adaptive search(GA) and fuzzy neural network(ANFIS). Backpropagation is the most popular algorithm for feed forward supervised learning. Backpropagation as well as its four variations, including radial basis function (RBF) are also developed. Genetic adaptive search is used mainly as an optimization tool to get faster and better convergence during training. The new algorithms for example RBF, GA+ANN ,adaptive network fuzzy inference system (ANFIS) and finally GA +ANFIS are developed. All the algorithms are applied to desulphurization and dephosphorization in iron and steel making. The performance of algorithms are tested on real plant data and compared on the basis of statistical performance parameters such as root mean square error and correlation coefficient during the training and testing phases. A special feature of this work is that minimal information about the theory of iron and steel making processes is used and all the training and testing is done on the raw data obtained from industry. GA +ANFIS produces the best generalized results for dephosphorization problem when just two rules are employed for each of the input parameters. As the number of parameters and rules increases, the problem of overlearning occurs in almost all the cases. Radial basis function is found to

be best alternative to the backpropagation provided the number of patterns for training is large. There is further scope for improvement of results if raw data are preprocessed through the theoretical model of the desulphurization and dephosphorization and then used for training.

CONTENTS

| | |
|---|---------------|
| LIST OF FIGURE | viii |
| LIST OF TABLES | ix |
| 1 Introduction | 1 |
| 1.1 Introduction to Neural Networks | 1 |
| 1.1.1 Input and output patterns | 2 |
| 1.1.2 Connections | 3 |
| 1.1.3 Neurons/Processing Elements | 4 |
| 1.2 Integration of Genetic algorithm and Fuzzy Logic with neural networks | 5 |
| 1.3 Problem Description | 5 |
| 1.3.1 Dephosphorization Problem | 5 |
| 1.3.2 Desulphurization Problem | 6 |
| 1.4 Organization of thesis | 7 |
| 2 FEED FORWARD ARTIFICIAL NEURAL NETWORK PERCEPTRON AND RADIAL BASIS FUNCTION MODELS | 8 |
| 2.1 : Introduction to backpropagation | 9 |
| 2.1.1 Backpropagation Network | 9 |
| 2.1.2 Activation Function | 10 |
| 2.1.3 Backpropagation algorithm | 12 |
| 2.2 : Limitations of backpropagation | 20 |
| 2.3 Variations of Backpropagation | 22 |
| 2.3.1 Variation 1 : Random presentation of patterns | 22 |
| 2.3.2 Variation 2 : Proportionate error learning | 25 |
| 2.3.3 Variation 3 : Logarithmic Error Function | 27 |
| 2.3.4 Variation 4 :Logarithmoid activation function | 29 |
| 2.4 : Radial Basis Function | 32 |
| 2.4.1 K-mean Clustering algorithm | 35 |
| 2.4.2 Determination of scaling factor or width (σ) of RBF | 36 |
| 2.4.3 Calculation of hidden layer output $A_{k,j}$ for for all patterns | 37 |
| 2.4.4 : Training the weights in second layer | 38 |
| 3 Integration of ANN with Intelligent Paradigms | 40 |
| 3.1 Backpropagation using Genetic Algorithm | 40 |
| 3.1.1 Learning Procedure with GA | 41 |
| 3.2 GA assisted neural network plus backpropagation combined | 46 |
| 3.3 Fuzzy Neural Network | 47 |
| 3.3.1 Fuzzy if- then rules and fuzzy inference system | 49 |
| 3.3.2 Architecture of Generalized Adaptive Network and learning algorithm | 52 |
| 3.4 GA + ANFIS | 67 |

| | |
|---|------------|
| Chapter 4 :Results and discussion | 69 |
| 4.1 Results of backpropagation and its variations | 70 |
| 4.1.1 : Dephosphorization problem | 70 |
| 4.1.2 : Desulphurization problem | 73 |
| 4.2 Results of Radial Basis Function | 74 |
| 4.2.1 :An example of RBF learning algorithm on Dephosphorization problem | 74 |
| 4.2.2 : Results of dephosphorization problem | 81 |
| 4.2.3 : Results of RBF on desulphurization problem | 82 |
| 4.3 Results of GA assisted BP (GA + BP) continued with BP | 82 |
| 4.3.1 : Dephosphorization problem | 83 |
| 4.3.2 : Desulphurization problem | 84 |
| 4.4 Results of ANFIS | 85 |
| 4.4.1 An example of learning of ANFIS network of type Fig (3.4 a) | 85 |
| 4.4.2 : Dephosphorization problem | 91 |
| 4.4.3 : Desulphurization problem | 92 |
| 4.5 Results of GA assisted ANFIS | 93 |
| 4.5.1 : Dephosphorization problem | 94 |
| 4.5.2 : Desulphurization problem | 94 |
| 4.6 Comparison of results of various algorithm | 95 |
| 5 Conclusions and Recommendations for future work | 98 |
| 5.1 Conclusions : | 98 |
| 5.2 Recommendation for future work : | 100 |
| References | 102 |
| Appendix A | 104 |
| Tables | 110 |

LIST OF FIGURES

| FIGURE NUMBER | TITLE |
|---------------|---|
| 1.1 | Simple neural network |
| 1.2 | A simple neuron |
| 2.1 | Backpropagation Network |
| 2.2 | Sigmoidal function |
| 2.3 | Showing local minima |
| 2.4 | Logarithmoid activation function |
| 2.5 | RBF Network |
| 2.6 | Guassian Kernel |
| 3.1 | Arrangement of weights in chromosome |
| 3.2 | Fuzzy inference system |
| 3.3 | Adaptive Network |
| 3.4 | ANFIS Networks |
| 4.1 | Learning plot for different variations backpropagation |
| 4.2 | Learning plot for different algorithm |
| A1 | Cycle of Genetic algorithms |

LIST OF TABLES

| TABLE NO. | TITLE |
|-----------|--|
| 1.1 | Learning data used for dephosphorization problem |
| 1.2 | Testing data used for dephosphorization problem |
| 1.3 | Learning data used for desulphurization problem |
| 1.4 | Testing data used for desulphurization problem |
| 4.1 | Minimum and maximum values of scaling for Dephosphorization data |
| 4.2 | Minimum and maximum values of desulphurization data |
| 4.3 | Results of Backpropagation on Dephosphorization Problem : |
| 4.4 | Results of Backpropagation - Variation 1 on Dephosphorization Problem |
| 4.5 | Results of Backpropagation - Variation 2 on Dephosphorization Problem |
| 4.6 | Results of Backpropagation - Variation 3 on Dephosphorization Problem |
| 4.7 | Results of Backpropagation - Variation 4 on Dephosphorization Problem (on scaled data) |
| 4.8 | Results of Backpropagation - Variation 1 on Dephosphorization Problem (unscaled data) |
| 4.9 | Results of Backpropagation on Desulphurization Problem : |
| 4.10 | Results of Backpropagation - Variation 1 on Desulphurization Problem |
| 4.11 | Results of Backpropagation - Variation 2 on Desulphurization Problem |
| 4.12 | Results of Backpropagation - Variation 3 on Desulphurization Problem |
| 4.13 | Results of Backpropagation - Variation 4 on Desulphurization Problem (scaled data) |
| 4.14 | Results of RBF on Dephosphorization |
| 4.15 | Results of RBF on Desulphurization |
| 4.16 | Results of GA + BP on Dephosphorization |

- 4.17 Results of GA+ BP on
desulphurization
- 4.18 Results of ANFIS (Network of type
Fig (3.4 a)) on dephosphorization
- 4.19 Results of ANFIS (Network of type
Fig (3.4 a)) on desulphurization
- 4.20 Results of ANFIS (Network of type
Fig (3.4 b)) on desulphurization
- 4.21 Results of GA assisted ANFIS
(Network of type Fig (3.4 a))
on dephosphorization
- 4.22 Results of GA assisted ANFIS
(Network of type Fig (3.4 a))
on desulphurization problem
- 4.23 Results of GA assisted ANFIS
(Network of type Fig (3.4 b))
on desulphurization problem
- 4.24 Comparison of all algorithms on
Dephosphorization Problem

CHAPTER 1

INTRODUCTION

One of the advantages of neural networks is that they can be used as a modeling tool without using detailed knowledge of a process. In iron and steel industry, in particular, most of the processes are difficult to model perfectly by classical techniques. It has now been found that neural networks could be used to model these processes with or without the aid of models already available.

1.1 Introduction to Neural Networks

Neural networks are essentially information processing systems with an adaptability and ability to learn. Some of the other names given to neural networks are artificial neural systems, connectionist models, parallel distributed processing models, layered self adaptive systems, self organizing systems, neurocomputers, neuromorphic systems and cyberware. In this work they are simply referred to as neural networks.

Historically, neural network was originally conceived as an attempt to model the biophysiology of human brain [1], i.e. to understand and explain how the brain operates and functions. The idea, though far fetched and nowhere close to reality, was to make a model which mimics a human brain. As things stand today neural networks can be thought of as "black box" device that

accepts inputs and produces outputs with an ability to learn the mappings between input variables and output feature spaces.

The two physical elements of the neural networks are: connections and neurons/processing elements [2] (Fig (1.1)) whereas the threshold function and input/output patterns constitute the operating elements. Each of these elements are described below one by one.

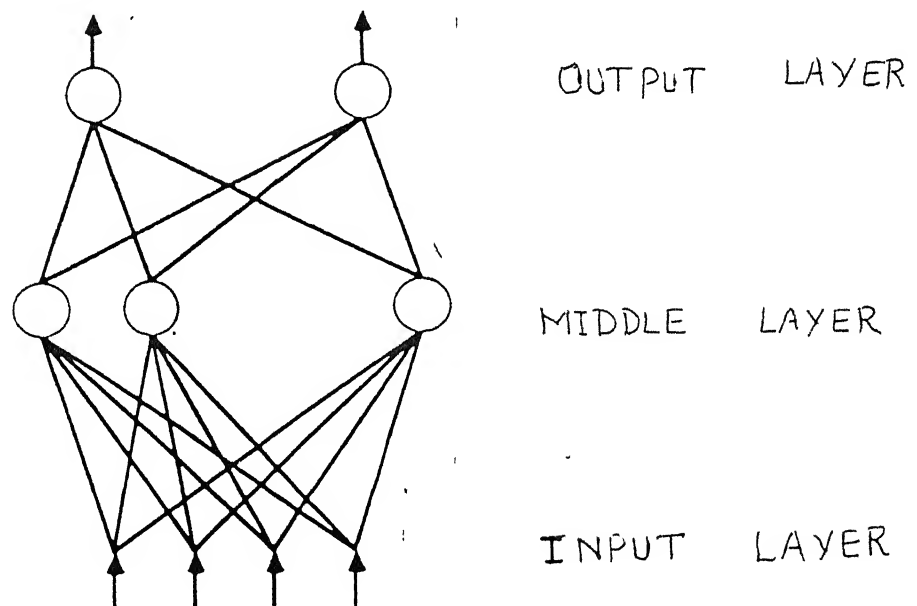


Fig (1.1) : Simple neural network [7]

1.1.1 Input and output patterns

Neural network can not operate until they have data.

Some neural networks require only single pattern (input pattern) , and others require pattern pairs (both input and output pattern). If only input pattern work is used, it is called an autoassociative network and when pattern pairs are used , it is called heteroassociative network. During modeling with neural networks ,it is necessary to specify the structure of pattern because the proper selection and representation of patterns greatly decides the performance of the network. In model based neural networks, the patterns can be preprocessed depending upon the function of relationships pertaining to the process. In model free networks, (with or without any modification) raw data are directly fed to the network.

1.1.2 Connections

A neural network is equivalent to directed graph (digraph). A digraph has edges (connections) between nodes (neurons) that allow information to flow only in one direction (the direction denoted by the arrow (Fig (1.1)) . Information flows through the digraph along the edges and is collected at the nodes. In contrast to digraph , the connections in the neural network define the direction as well as weight of information (through connection weights). Connection weights are adjusted by a learning algorithm which helps to capture process information. Positive weights are excitatory connections. and negative weights are inhibitory connections. A connection with zero weight is equivalent to not having the particular connection at all.

1.1.3 Neurons/Processing Elements

Neurons (or nodes) are the sites where all computing is performed. Fig (1.2) illustrates the most common type of neuron. Each neuron collects the information from it's abutting connections and produces a single output value.

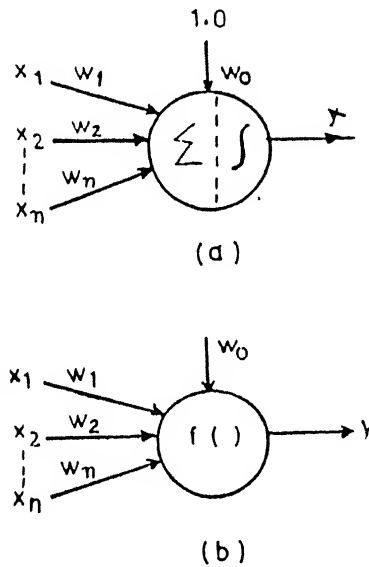


Fig (1.2) : A simple neuron

It is suggested to integrate different artificial intelligence (AI) techniques with neural networks for better performance [3].

1.2 Integration of Genetic algorithm and Fuzzy Logic with neural networks

Learning algorithm most often requires an optimization method. Usually in neural networks gradient based methods are used. Genetic algorithms are evolutionary optimization methods which can be used as an alternative to gradient based method for better performance [3].

Fuzzy logic [4] is used to deal with linguistic uncertainty in different processes. It is possible to incorporate neural networks with fuzzy logic [5]. Thus in fuzzy neural networks fuzzy neurons can be used where different *and/or* operation can be done.

1.3 Problem Description

Dephosphorization of steel produced in combined blown converter and desulphurization of hot metal in torpedo are the two problems which are modeled in this work.

1.3.1 Dephosphorization Problem

The hot metal charged in the converter contains carbon, silicon, manganese, sulphur, and phosphorus. The final phosphorus content of steel at the end of blow can be considered to be a

function of seven input variables , namely hot metal weight (in tons), hot metal temperatures (in degree C), C ,Mn and P contents of hot metal (in 100 x %age), amount of slag and, lastly, the amount of oxygen blown during the heat. These seven inputs variables constitute seven input nodes in the input layer and the single output is the final phosphorus content of steel. In order to decide the network parameters and training strategy for a network so as to make the net conversant with as many as patterns as possible, it was decided to train the network with 200 patterns of input - output combinations. Testing of the net was done with 96 set of input combinations to predict final phosphorus content. The output was compared with actual phosphorus content to find prediction error. These data sets for the training and testing are provided in Tables (1.1,1.2).

1.2.2 Desulphurization Problem

In this problem the final sulphur content of hot metal during calcium carbide treatment is predicted as a function of four inputs, namely initial sulphur content, hot metal temperature, hot metal weight and amount of calcium dinamide ($\text{CaC}_2 + \text{CaO}$) injected. Carrier gas flow rate is kept constant. Altogether 350 samples are used for training and then the network is tested with 161 unseen samples. Both training and testing patterns are provided in Tables (1.3,1.4).

1.4 Organization of thesis

The backpropagation including, its four variations and the Radial Basis function are discussed in chapter 2. Integration of Genetic algorithm and fuzzy logic with neural networks is discussed in chapter 3. The results of all the algorithms (discussed in chapter 2 and 3) with respect to the two problems of dephosphorization and desulphurization are discussed in chapter 4. Finally the conclusions and future scope of the work is presented in chapter 5.

CHAPTER 2

FEED FORWARD ARTIFICIAL NEURAL NETWORK PERCEPTRON AND RADIAL BASIS FUNCTION MODELS

Backpropagation (BP) algorithm of training is the most popular method to map a set of inputs to a set of outputs among the current types of feed forward supervised neural networks. It has thus played an important role in the resurgence of interest in artificial neural networks [1]. The convergence of the BP is, however, known to be slow, i.e. it may require a large training time. Radial Basis Function (RBF) is a good alternative to BP because it requires fewer weights and also needs much less training time.

In the recent past many variations of BP have been proposed to reduce training time [7]. In the present work, the performance of some of the variations of BP and RBF are compared. Backpropagation is introduced in section 2.1 followed by the discussion of the limitations of BP in section 2.2. The variations of BP are discussed in section 2.3. Finally RBF is discussed in section 2.4.

2.1 : Introduction to backpropagation

Backpropagation with Generalized Delta Rule (GDR) was initially developed by Rumelhart, Hinton and William in 1986 [1]. Since then it has dramatically expanded the range of problems to which artificial neural network could be applied to [1,8,9]. The elements of BP are discussed below.

2.1.1 : Backpropagation Network

The BP network (Fig 2.1) is an extension of the Multiple Adaptive Linear Element (MADALINE) structure [1] involving the summation and non linear activation function on each neuron and the use of multiple adaptive layers. In most applications, BP network usually consists of three types of layers : an input layer , a hidden layer(s) , an output layer. As shown in Fig (2.1) in the first layer (input layer) nodes/neurons are connected to external (or environmental) data. The second layer (or hidden layer(s)) consists of processing nodes/neurons which transmit the results of computation to the output layer. The number of neurons in the input layer is equal to the number of inputs in each pattern , and each of these neurons receive one of the inputs. The outcome of each of the neurons in the output layer is the actual output of the network. The choice of number of layers and neurons in each hidden layer(s) is problem dependent. The neurons between two consecutive layers are connected with links. There are no interconnections/links within neuron in the same layer. The links connecting the neurons have weights

associated with them. Each neuron in a layer provides an input to each and every neuron on the next layer e.g. each neuron in input layer sends its output to every neuron in the next hidden layer. BP has been designed to adjust the weights by following an iterative procedure in such a manner that a desired result can be predicted by training on a set of representative examples (or patterns) of a process. The basic components of the algorithm are as follows :

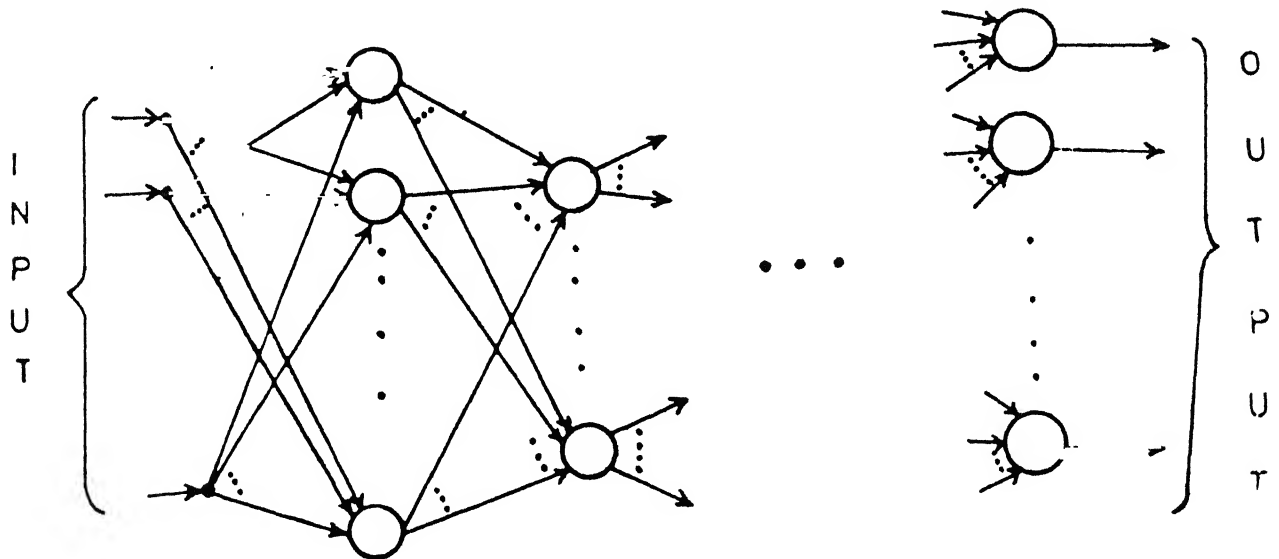


Fig (2.1) : Backpropagation Network

2.1.2 Activation Function

BP has been developed to recognize any nonlinear relation between input and output. In the absence of nonlinearity BP would implement nothing more than a linear transformation at each layer, in which case it (BP) could be reduced to an

equivalent single layer network. In BP the nonlinear function evaluated at each neuron in middle layer(s) and output layer, is known as activation function. Sigmoidal function is generally used as an activation function; it is continuous, 'S' shaped, monotonically increasing, continuously differentiable, and asymptotically approaches fixed finite values as the input approaches plus or minus infinity [7]. The fixed asymptotic values are generally zero and +1 or -1 and +1. An example of the sigmoidal function (Fig 2.2) is :

$$f(x) = \frac{1}{1 + \exp(-x / g)} \quad \dots(2.1)$$

where $g = \text{logistic/gain}$

The effect of t is to modify the shape of sigmoidal function. Other examples of activation function are

(i) Hyperbolic tangent

$$f(x) = \tanh(x) = \frac{\frac{e^x}{x} - \frac{e^{-x}}{-x}}{\frac{e^x}{x} + \frac{e^{-x}}{-x}} \quad \dots(2.2)$$

with limiting values of -1 to 1.

(ii) Ratio of squares

$$\begin{aligned}
 f(x) &= \frac{x^2}{1 + x^2} \quad , \text{ if } x > 0 \\
 &= 0 \quad , \text{ if } x \leq 0 \quad \dots (2.3)
 \end{aligned}$$

with limiting 0 to 1.

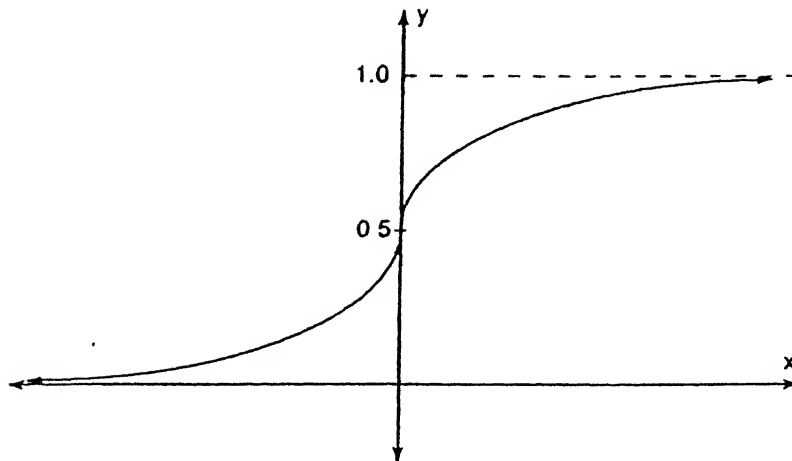


Fig (2.2) : Sigmoidal function

2.1.3 : Backpropagation algorithm

Backpropagation network typically begins with a random set of small weights. The network then adjusts the weights each time it processes an input-output pair. Correction of weight in each pair requires two stages : a forward pass and a backward pass. In forward pass outcome from input data is generated by

flowing the signals in forward direction sequentially from input to hidden and then from hidden to output layer as follows. The given set of input is imposed on the neurons in the input layer. These neurons transmit the resulting values to neurons in the hidden layer. The neurons in the hidden layer individually product the weights and input signals from previous layer and transform it with the help of activation function and then transmit the output signal to the next layer. The process continues at each layer until it reaches the output layer. The outcome of the neurons in the output layer is actual output generated by the network. During the backward pass, the network's actual output (from the forward pass) is compared with the target output and error estimates are computed for the output units. The weights connected to the output units can be adjusted in order to reduce the errors. The error estimates of the output units can be used to derive the error estimates for the units in the hidden layers. Finally, errors are propagated back to the connections stemming from the input units. Since the error is propagated backwards, this algorithm is known as backpropagation [7].

Let the basic elements of the network be denoted as follows :

Number of Patterns = P

Number of Layers = L

Number of neurons in different layers = $n(l)$

$l=1..L$

If bias neuron is added then it is equivalent to adding one neuron in each layer (except in output layer).

Thus number of inputs = $n(1)$

and number of outputs = $n(L)$

Input matrix = $x_{k,j}$

$k = 1 \dots P$

$j = 1 \dots n(1)$

Output matrix = $y_{k,j}$

$k = 1 \dots P$

$j = 1 \dots n(L)$

Weight connection from the i^{th} neuron of the l^{th} layer to the j^{th} neuron of the $(l+1)^{th}$ layer is given by $w_{l,i,j}$

$l = 1 \dots (L-1)$

$i = 1 \dots n(l) + 1$

$j = 1 \dots n(l+1)$

For k^{th} data set, output of the j^{th} neuron in the l^{th} layer is given by $O_{k,l,j}$

$k = 1 \dots P$

$l = 1 \dots L$

$j = 1 \dots n(l)$

Error in the k^{th} set is given by : E_k

Total Error in all patterns = E

Momentum factor = α

Learning rate = β

Logistic = g

Error signal = $\delta_{l,j}$

$l = 1 \dots L$

$j = 1 \dots n(l)$

Bias parameter = $B \{ =1 \text{ if net uses bias neuron else } =0 \}$

Significance of terms E_k , E , α , β , g , $\delta_{i,j}$ and B are explained later. The algorithm comprises essentially of seven steps

- Step (i) : Preprocess or transform (scale) the data.
- Step (ii) : Generate the initial random weights.
- Step (iii) : Evaluate the network output for a data set .
- Step (iv) : Calculate the error signal.
- Step (v) : Modify the weights by calculating the error derivative.
- Step (vi) : Repeat from step (iii) to (v) for all data sets.
- Step (vii) : Repeat step (iii) to (vi) until convergence.

Details of each of the step are as follows :

Step (i) Data Preprocessing

Activation function (sigmoidal function) is sensitive between small ranges (Fig 2.2) and it's output varies between 0 and 1. Since sigmoidal function is insensitive near zero or 1, it is suggested that both input and output should be linearly scaled between 0.1 and 0.9 to obtain better performance . The formula used for normalization is :

$$x_{scale} = 0.1 + 0.8 * \frac{x_{unscale} - x_{min}}{x_{max} - x_{min}} \quad \dots (2.4)$$

where x_{min} and x_{max} are the minimum and maximum, within which the values are to be scaled ; $x_{unscale}$ is the raw value and x_{scale} is the normalized value.

Step (ii) Starting Weights

Initial weights are chosen randomly between -0.5 to 0.5.

Step (iii) Evaluation of network output

Output of the input layer is equal to the input pattern. So for the k^{th} pattern learning the output is calculated by the following equations : For the first layer :

$$O_{k,1,j} = x_{k,j} \quad \dots (2.5)$$

$$k = 1 \dots P$$

$$j = 1 \dots n(1)$$

For the bias neuron the output is set equal to unity, i.e.

$$O_{k,1,n(1)+1} = 1 \quad \dots (2.6)$$

$$k = 1 \dots P$$

$$l = 1 \dots (L-1)$$

Output of the consequent layers are calculated by :

$$O_{k,l,j} = f \left[\sum_{i=1}^{(n(l-1) + B)} O_{k,l-1,i} w_{l-1,i,j} \right] \dots (2.7)$$

where f is a sigmoidal function and is given by Eq. (2.1) .

Step (iv) : Calculation of error in output

Let $O_{k,L,j}$ be the final output of the network while actual target is $y_{k,j}$. Actual least square error is then given by :

$$E = \sum_{k=1}^P \sum_{j=1}^{n(L)} \left[y_{k,j} - O_{k,L,j} \right]^2 \dots (2.8)$$

Now, the problem is to find the weights (w's), such that error is minimum.

Step (v) : Calculation of change in weight for k^{th} pattern

$$(\Delta w_{l,i,j})_k = -\beta * \left(\partial E / \partial w_{l,i,j} \right) + \alpha * (\Delta w_{l,i,j})_{k-1} \dots (2.9)$$

$$l = 1 \dots (L-1)$$

$$i = 1 \dots n(l)$$

where β is the learning rate and α is momentum factor. Error derivative with respect to weights $(\partial E / \partial w_{1,i,j})$ are given by the equation :

$$\partial E / \partial w_{1,i,j} = \delta_{1+1,j} O_{k,1+1,j} \quad \dots (2.10)$$

where $\delta_{1+1,j}$ is given by Eq. (2.11,2.12). The following equation gives the expression for δ for the last layer i.e. the output layer.

$$\delta_{L,j} = 2 * (Y_{k,j} - O_{k,L,j}) * f'(O_{k,L,j}) \quad \dots (2.11)$$

$$j = 1 \dots n(L)$$

and $\delta_{1,j}$ for the previous layer is given by Generalized Delta rule [7] :

$$\delta_{1,j} = \sum_{i=1}^{n(1+1)} (\delta_{i,j} * w_{1,i,j}) * f'(O_{k,1,j}) \quad \dots (2.12)$$

$$l = 1 \dots L-1$$

$$j = 1 \dots n(l)$$

f' (in Eq.(2.11,2.12)) is the derivative of processing element function and for sigmoidal function it is given by :

$$f'(x) = f(x) * (1 - f(x)) / g$$

$$\text{or } f'(O_{k,L,j}) = O_{k,L,j} * (1 - O_{k,L,j}) / g \quad \dots (2.13)$$

The pseudo code of BP is provided below :

Procedure Backpropagation

Begin

Initialize weights randomly between -0.5 to +0.5

Initialize change in weight Δw equals to zero

for t = 1 to maxiteration do

begin

/* for All Patterns do */

for k = 1 to P do

begin

/* Calculate Output for each neuron */

for j = 1 to n(1) do

$$O_{k,1,j} = x_{k,j}$$

/* For bias neuron Output is set equals to unity */

for l = 1 to L-1 do

$$O_{k,1,n(1)+1} = 1.0$$

for l = 2 to L do

for j = 1 to n(l) do

$$O_{k,l,j} = f \left[\sum_{i=1}^{n(l-1)+B} O_{k,l-1,i} w_{l-1,i,j} \right]$$

/* $f(x) = 1/(1 + \exp(-x/g))$ */

/* Calculate δ */

for j = 1 to n(L) do

$$\delta_{L,j} = 2 * (y_{k,j} - O_{k,L,j}) * f'(O_{k,L,j})$$

/* $f'(x) = O * (1 - O) / g$ where $O = f(x)$ */

for l = L-1 down to 1 do

for j = 1 to n(L) do

$$\delta_{l,j} = \sum_{i=1}^{n(l+1)} (\delta_{l+1,i} * w_{l,i,j}) * f'(O_{k,l,j})$$

/* Update weight */

for l = 1 to L-1 do

for i = 1 to n(L) do

for j = 1 to n(l+1) do

begin

$$\partial E / \partial w_{l,i,j} = \delta_{l+1,i} O_{k,l+1,j}$$

$$(\Delta w_{l,i,j})_k =$$

$$-\beta * (\partial E / \partial w_{l,i,j}) + \alpha * (\Delta w_{l,i,j})_{k-1}$$

$$w_{l,i,j} = w_{l,i,j} + (\Delta w_{l,i,j})_k$$

end;

end; /* for all patterns */

end; /* for all iteration */

End /* Procedure Backpropagation */

2.2 : Limitations of backpropagation

Despite of many successful applications of BP, it is not a panacea. Most troublesome is the uncertain and long training process which can arise as a result of a non optimum choice of learning parameters, for example :learning rate and momentum factor. Three important limitations of BP are :

(i) Network paralysis

In some cases , as the training progresses the weights may become adjusted to very large values. This causes the aggregate of inputs to a neuron to become extremely large (either +ve or -ve) and hence it approaches the saturation region of sigmoidal function. The derivative of error with respect to weights in saturation region of sigmoidal function is ,evidently, small. Therefore the corresponding weight correction ($\Delta w_{1,i,j}$) will also be small and hence the training will be extremely slow [10].

(ii) Local minima

Backpropagation employs a type of gradient descent

though the solution is not close to the optimal solution. In such a case the activation function becomes insensitive when the output approaches 0 or 1 [20]. In order to reduce the training time, several variations of BP, in particular the method based on Newton's method and second derivative have been tried by various workers [22] .

2.3 Variations of Backpropagation

There are numerous variations of BP have been suggested [10]. In the present work following four variations of BP have been tried .

1. Random presentation of patterns
2. Proportionate error learning
3. Logarithmic Error Function
4. Logarithmoid activation function

The details of each of the four variations are given below.

2.3.1 Variation 1 : Random presentation of patterns

The learning weights in BP are usually updated by sequential presentation of training patterns. In this scheme one training iteration incorporates one sweep through all training patterns. Such an iteration scheme is called an epoch . The

complete training requires many epochs. Training patterns can also be presented in a random fashion [11]. The idea behind the random presentation of patterns is to break the sequence of pattern learning and ,therefore, enhance the speed of convergence and the generalization ability of the network. The steps involved are as follows :

- Step (i) : Preprocess or transform (scale) the data.
- Step (ii) : Generate the initial random weights.
- Step (iii) : Evaluate the network output for a data set .
- Step (iv) : Calculate the error signal.
- Step (v) : Modify the weights by calculating the error derivative.
- Step (vi) : Repeat from step (iii) to (v) for all data sets.
- Step (vii) : Randomly shuffle the order of patterns .
- Step (viii) : Repeat step (iii) to (vii) until convergence.

Thus only the step (vii) needs to be added to the algorithm of BP. The pseudo code for the variation is given below:

Procedure Backpropagation_variation1

Begin

```

Initialize weights randomly between -0.5 to +0.5
Initialize change in weight  $\Delta w$  equals to zero
/* Initialize an one dimension array r */

```

```

for i = 1 to P do
    ri = i
for t = 1 to maxiteration do
begin
    Shuffle elements in array r
    /* for All Patterns do */
    for m = 1 to P do

begin
    k= rm
    /* Calculate Output for each neuron */
    for j = 1 to n(1) do
        Ok,1,j = xk,j

    /* For bias neuron Output is set equals to unity */
    for l = 1 to L-1 do
        Ok,1,n(1)+1 = 1.0
    for l = 2 to L do
        for j = 1 to n(l) do
            
$$O_{k,l,j} = f \left[ \sum_{i=1}^{(n(l-1)+B)} O_{k,l-1,i} w_{l-1,i,j} \right]$$

            /* f(x) = 1/(1+ exp (-x/g)) */
            /* Calculate  $\delta$  */
            for j = 1 to n(L) do
                
$$\delta_{L,j} = 2 * (y_{k,j} - O_{k,L,j}) * f'(O_{k,L,j})$$

            /* f'(x) = O * (1- O) /g where O = f(x) */
            for l = L-1 down to 1 do
                for j =1 to n(L) do
                    
$$\delta_{l,j} = \sum_{i=1}^{n(l+1)} (\delta_{l+1,i} * w_{l+1,i,j}) * f'(O_{k,l,j})$$


            /* Update weight */
            for l = 1 to L-1 do
                for i = 1 to n(L) do
                    for j = 1 to n(l+1) do
begin
                        
$$\partial E / \partial w_{l,i,j} = \delta_{l+1,j} O_{k,l+1,j}$$

                        
$$(\Delta w_{l,i,j})_k =$$

                        
$$-\beta * (\partial E / \partial w_{l,i,j}) + \alpha * (\Delta w_{l,i,j})_{k-1}$$

                        
$$w_{l,i,j} = w_{l,i,j} + (\Delta w_{l,i,j})_k$$

                    end;
                end; /* for all patterns */
            end; /* for all iteration */
        End /* Procedure Backpropagation_variation1 */

```

2.3.2 Variation 2 : Proportionate error learning

In the proportionate error learning , the number of times a pattern is presented is made proportional to the error value . According to this scheme each pattern at each epoch is presented for learning $\lceil (E_k * P / \sum E_k) \rceil$ number of times as compared to only once in BP; E_k is the error in k^{th} pattern and P is number of patterns. Thus, the patterns which have comparatively higher error are presented more number of times than those which have lower error. The idea is to enhance the learning speed. The algorithm can be given in following steps :

- Step (i) : Preprocess or transform (scale) the data.
- Step (ii) : Generate the initial random weights.
- Step (iii) : Evaluate the network output for a data set .
- Step (iv) : Calculate the error signal.
- Step (v) : Modify the weights by calculating the error derivative.
- Step (vi) : Repeat (iii) to (v) $\lceil E_k * P / \sum E_k \rceil$ no of times.
- Step (vii) : Repeat from step (iii) to (vi) for all data sets.
- Step (viii) : Repeat step (iii) to (vii) until convergence.

The only step (vi) needs to be added to BP. The pseudo code for the variation is given below :

Procedure Backpropagation_variation2

Begin

Initialize weights randomly between -0.5 to +0.5
 Initialize change in weight Δw equals to zero
 /* Initialize an array r */


```

for i = 1 to P do
  ri = 1
for t = 1 to maxiteration do
begin
  /* for All Patterns do */
  for k = 1 to P do
    for m = 1 to rk do
      begin
        /* Calculate Output for each neuron */
        for j = 1 to n(1) do
          Ok,1,j = xk,j

          /* For bias neuron Output is set equals to unity */
          for l = 1 to L-1 do
            Ok,1,n(1)+1 = 1.0
          for l = 2 to L do
            for j = 1 to n(l) do
              
$$O_{k,l,j} = f \left[ \sum_{i=1}^{(n(l-1)+B)} O_{k,l-1,i} w_{l-1,i,j} \right]$$

              /* f(x) = 1/(1+ exp (-x/g)) */
              /* Calculate  $\delta$  */
              for j = 1 to n(L) do
                
$$\delta_{L,j} = 2 * (y_{k,j} - O_{k,L,j}) * f'(O_{k,L,j})$$

              /* f'(x) = O * (1- O) /g where O = f(x) */
              for l = L-1 down to 1 do
                for j = 1 to n(l) do
                  
$$\delta_{l,j} = \sum_{i=1}^{n(l+1)} (\delta_{l+1,i} * w_{l+1,i,j}) * f'(O_{k,l,j})$$


              /* Update weight */
              for l = 1 to L-1 do
                for i = 1 to n(l) do
                  for j = 1 to n(l+1) do
                    begin
                      
$$\partial E / \partial w_{l,i,j} = \delta_{l+1,j} O_{k,l+1,j}$$

                      
$$(\Delta w_{l,i,j})_k =$$

                      
$$-\beta * (\partial E / \partial w_{l,i,j}) + \alpha * (\Delta w_{l,i,j})_{k-1}$$

                      
$$w_{l,i,j} = w_{l,i,j} + (\Delta w_{l,i,j})_k$$

                    end;
                  end; /* for all patterns */
                /* Find new array r */
                for i = 1 to P do

```

```

         $r_i = E_i * P / E$ 
    end; /* for all iteration and for r times loop */
End /* Procedure Backpropagation_variation2 */

```

2.3.3 Variation 3 : Logarithmic Error Function

The derivative of sigmoidal function, as explained earlier in section 2.1.3 contains $O(1-O)$ term (Eq.(2.13)), where O is the the output of neuron. The value of $O(1-O)$ term is very small when O approaches either zero or one . Since error derivative $\partial E / \partial w$ (Eq.(2.11)) consists of $O(1-O)$ term, the value of error derivative also becomes very small even when it is not close to optima. Therefore, the error derivative $\partial E / \partial w$ will be small even though the actual output produced by the network $O_{k,L,j}$, is not close to the target value $y_{k,j}$. Therefore the search for a minimum in the error is retarded. For instance, this occurs when $O_{k,L,j}$ approaches either zero or one (owing to the nature of derivative of the activation function (Eq.(2.13))) . Unfortunately any saturating response function is bound to have this property because near the saturation point the error derivative vanishes. Such a limitation can be overcome by using a slightly modified error function. Instead of minimizing the the square of difference between the actual $O_{k,L,j}$ and target $y_{k,j}$ values summed over the output units j , for all patterns k , the following error function has been suggested [12] :

$$E = - \sum_{k=1}^P \sum_{j=1}^{n(L)} \left[y_{k,j} \ln(o_{k,L,j}) + (1-y_{k,j}) \ln(1-o_{k,L,j}) \right] \quad \dots (2.14)$$

The derivative of E with respect to neuron output is given by

$$\frac{\partial E}{\partial o_{k,L,j}} = \frac{o_{k,L,j} - y_{k,j}}{o_{k,L,j} (1 - o_{k,L,j})} \quad \dots (2.15)$$

The derivative term in Eq.(2.15) contains $O(1-O)$ term in denominator which cancels out the $O(1-O)$ term in the numerator and is, thus, expected to remove the slowness in training. The only modification needed in BP is in Eq.(2.11) and is given by :

$$\delta_{L,j} = 2 * (y_{k,j} - o_{k,L,j}) / g \quad \dots (2.16)$$

The pseudo code for the variation is given below :

Procedure Backpropagation_variation3

Begin

Initialize weights randomly between -0.5 to +0.5

Initialize change in weight Δw equals to zero

for t = 1 to maxiteration do

begin

/* for All Patterns do */

for k = 1 to P do

. begin

/* Calculate Output for each neuron */

for j = 1 to n(1) do

$o_{k,1,j} = x_{k,j}$

/* For bias neuron Output is set equals to unity */

```

for l = 1 to L-1 do
  Ok,1,n(1)+1 = 1.0
for l = 2 to L do
  for j = 1 to n(l) do
    Ok,1,j = f [  $\sum_{i=1}^{(n(l-1)+B)}$  Ok,l-1,i wl-1,i,j ]
    /* f(x) = 1/(1+ exp (-x/g)) */
    /* Calculate  $\delta$  */
    for j = 1 to n(L) do
       $\delta_{L,j} = 2 * (-y_{k,j} + O_{k,L,j})$ 
    /* f'(x) = O * (1 - O) /g where O = f(x) */
    for l = L-1 down to 1 do
      for j = 1 to n(l) do
         $\delta_{l,j} = \sum_{i=1}^{n(l+1)} (\delta_{l+1,i} * w_{l,i,j}) * f'(O_{k,l,j})$ 

        /* Update weight */
        for l = 1 to L-1 do
          for i = 1 to n(L) do
            for j = 1 to n(l+1) do
              begin
                 $\partial E / \partial w_{l,i,j} = \delta_{l+1,j} O_{k,l+1,j}$ 
                 $(\Delta w_{l,i,j})_k =$ 
                 $-\beta * (\partial E / \partial w_{l,i,j}) + \alpha * (\Delta w_{l,i,j})_{k-1}$ 
                 $w_{l,i,j} = w_{l,i,j} + (\Delta w_{l,i,j})_k$ 
              end;
            end; /* for all patterns */
          end; /* for all iteration */
        End /* Procedure Backpropagation_variation3 */

```

2.3.4 Variation 4 : Logarithmoid activation function

Logarithmoid function [9] (Fig (2.4)) has been suggested as an alternative to sigmoidal function and is given by

$$f(x) = \frac{x}{|x|} \ln (1 + c |x|) \quad \dots (2.17)$$

where c = constant

Logarithmoid function (Eq.(2.17)), Fig (2.4)) is continuous, monotonically increasing from minus infinity to plus infinity ($-\infty$ to $+\infty$), continuously differentiable and is most sensitive near $x=0$. The basic advantage of using logarithmoid function is that it's output is not bound to any fixed values and it does not need any type of scaling for input and output patterns. The derivative of logarithmoid function is given by :

$$\begin{aligned}
 \partial f / \partial x &= c / (1 + c|x|) \\
 &= c * \exp(-x/|x| * f(x)) \\
 &= c * \exp(-|f(x)|) \quad \dots (2.18)
 \end{aligned}$$

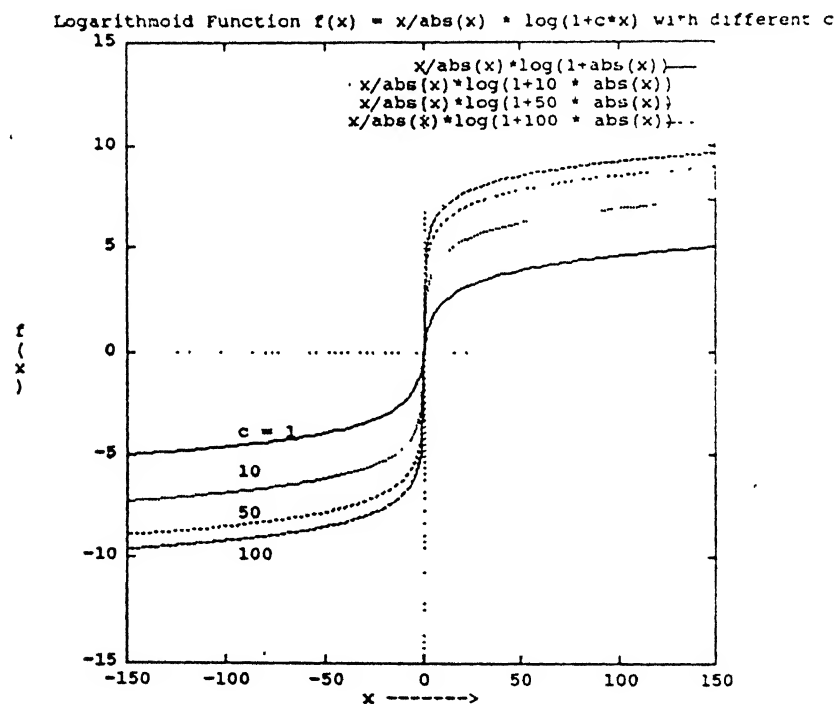


Fig (2.4): Logarithmoid activation function

Only modification needed in BP is to replace the

function in Eq.(2.7) with Eq.(2.17), and the derivative in Eq.(2.13) with Eq.(2.18). The pseudo code for the variation is given below :

Procedure Backpropagation_variation4

Begin

Initialize weights randomly between -0.5 to +0.5

Initialize change in weight Δw equals to zero

for t = 1 to maxiteration do

begin

/* for All Patterns do */

for k = 1 to P do

begin

/* Calculate Output for each neuron */

for j = 1 to n(1) do

$O_{k,1,j} = x_{k,j}$

/* For bias neuron Output is set equals to unity */

for l = 1 to L-1 do

$O_{k,1,n(1)+1} = 1.0$

for l = 2 to L do

for j = 1 to n(l) do

$$O_{k,l,j} = f \left[\sum_{i=1}^{(n(l-1)+B)} O_{k,l-1,i} w_{l-1,i,j} \right]$$

/* $f(x) = x/|x| * \ln(1/(1 + c * |x|))$ */

/* Calculate δ */

for j = 1 to n(L) do

$\delta_{L,j} = 2 * (y_{k,j} - O_{k,L,j}) * f'(O_{k,L,j})$

/* $f'(x) = c * \exp(-|O|)$ where $O = f(x)$ */

for l = L-1 down to 1 do

for j = 1 to n(L) do

$$\delta_{l,j} = \sum_{i=1}^{n(l+1)} (\delta_{l+1,i} * w_{l,i,j}) * f'(O_{k,l,j})$$

/* Update weight */

for l = 1 to L-1 do

for i = 1 to n(L) do

for j = 1 to n(l+1) do

begin

$\partial E / \partial w_{l,i,j} = \delta_{l+1,j} O_{k,l+1,j}$

```

      (  $\Delta w_{1,i,j}$  )k =
      - $\beta$  * (  $\partial E / \partial w_{1,i,j}$  ) +  $\alpha$  * (  $\Delta w_{1,i,j}$  )k-1
       $w_{1,i,j}$  =  $w_{1,i,j}$  + (  $\Delta w_{1,i,j}$  )k
    end;
  end; /* for all patterns */
end; /* for all iteration */
End /* Procedure Backpropagation_variation4*/

```

2.4 : Radial Basis Function

The incorporation of RBF in the neural network provides an alternative to BP for interpolation in a high dimension space. RBF requires fewer number of weights than in BP and possess good generalization ability. In addition to the input layer, RBF network comprises only of two more layers : one hidden layer, and one output layer [15]. The input nodes pass the input values to the connecting arcs with middle layer (Fig (2.5)), and connections between the input layer and middle layer are not weighted unlike in BP. Thus each hidden neuron directly receives unaltered input values . Each of The hidden neurons operates on a radial basis function (or transfer function) (Fig 2.6)). These functions in one dimension are of the form (Fig (2.6)) :

$$f(x) = \exp \left[- \frac{(x - c)^2}{\sigma^2} \right] \quad \dots (2.19)$$

where c is the center and σ is the width of radial basis function.

The value of c and σ are different for each neuron. On increasing σ , a given distance from center c , the output is large. Also, the range in which nonzero output is generated is large. These basis functions are also known as Gaussian kernel.

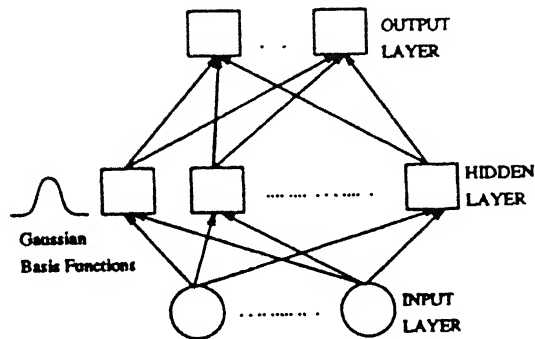


Fig 2.5 RBF Network [6]

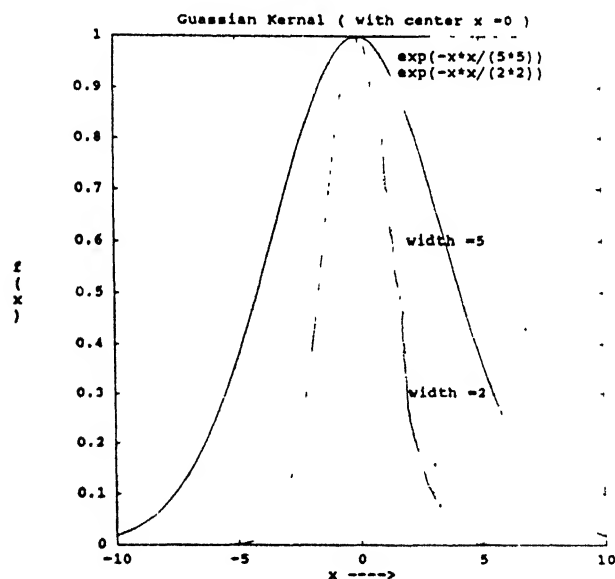


Fig (2.6) : Guassian Kernel

The RBF in Eq.(2.19) (Fig (2.6)) is nonmonotonic in

contrast to monotonic sigmoidal function used in BP . The connection in the second layer are weighted . The output of the network from an output neuron is the weighted sum of the product of output of hidden neuron and the corresponding weights to the particular output neuron . Similar to the case of perceptron a bias neuron (threshold) can be added in the hidden layer.

Theoretically, RBF like BP, is capable of performing an arbitrary close approximation of any continuous nonlinear mapping between input and output. The primary difference between RBF and BP is in the nature of their transformation or activation function used. As said earlier , the hidden layer neurons in BP use sigmoidal function (Eq.(2.1)) which are nonzero over an infinitely large region of input space , while the transformation function in RBF network (Eq.(2.19)) covers only small localized regions around it's center c . The RBF's are radially symmetric and produce nonzero output for inputs that lie within a small radial distance from it's center $(x-c)$.

Learning process of RBF network occurs in two stages : learning in the hidden layer, followed by learning in the output layer. Learning in the hidden layer is typically performed using an unsupervised method i.e. a clustering algorithm, while learning in the output layer is supervised; the most popular clustering algorithm is K-mean algorithm. (described in section 2.4.1). Terminology used for RBF is given below :

Number of input = n

Number of hidden neuron = h

Number of Output = m

Number of Patterns = P

Input matrix = $x_{k,j}$

$k = 1 \dots P$

$j = 1 \dots n$

Output matrix = $y_{k,j}$

$k = 1 \dots P$

$j = 1 \dots m$

Weights between hidden and output layer = $w_{i,j}$

$i = 1 \dots (h + 1)$

$j = 1 \dots m$

Width of Clustering Centers (for each basis function) = σ_i

$i = 1 \dots h$

Hidden layer Output matrix = $A_{i,j}$

$i = 1 \dots P$

$j = 1 \dots (h+1)$

2.4.1 K-mean Clustering algorithm

K-mean clustering algorithm finds a set of cluster centers and partitions the training data into subsets. For the sake of continuity, the subsets overlap to a limited extent. Each cluster center is associated with one of the hidden units (h) in the network. The data are partitioned such that the training points are assigned to the cluster with nearest center. A worked

out example using actual plant data is given in chapter 4. The pseudo Code for the K-mean algorithm is given below :

Procedure K_Mean

```

Begin
  Generate h different number between 1 to P and store it in
  array  $r_i$ 

  /*  $i = 1 \dots h$  and  $r$  is a storage array */

  /* Initialize Clustering Centers */
  for  $i = 1$  to  $h$  do
    for  $j = 1$  to  $n$  do
       $\hat{x}_{i,j} = x_{r_i,j}$ 
    /* Find Centers until there are no change in clustering
    centers */
  Repeat
    /*  $z$  is a 2 -d matrix for calculating new centers */
    /*  $s$  is a vector to store the number of patterns which
    fall around the chosen center */
    for  $i = 1$  to  $h$  do
      for  $j = 1$  to  $n$  do
        begin
           $s_i = 0; \quad z_{i,j} = 0;$ 
        end
      for  $i = 1$  to  $P$  do
        begin
          find  $m^{th}$  center which is closest to  $i^{th}$ 
          pattern ;
           $s_m = s_m + 1$ 
          for  $j = 1$  to  $n$  do
             $z_{m,j} = z_{m,j} + x_{i,j}$ 
          end;
          /* Calculate new centers */
          for  $i = 1$  to  $h$  do
            for  $j = 1$  to  $n$  do
               $(\hat{x}_{i,j})_{new} = (z_{i,j}) / s_i$ 
            Until ( Small changes in  $\hat{x}$  )
          End;
        /* End K_mean Procedure */

```

2.4.2 : Determination of scaling factor or width (σ) of RBF

Once the cluster centers are established, the receptive width of each function unit can be determined . The aim of setting the width of the RBF function is to cover the training points to allow a smooth fit of the desired network outputs. To achieve this , each hidden unit must activate at least one more unit to a significant degree. Therefore, RBF width is selected in such a way that it is greater than its distance from the nearest basis function center. However, the value of RBF width should be calculated such that there is no significant interference between working of centers which are far away from it. An appropriate width is determined from the nearest neighbor heuristic in a multidimensional space [15]:

$$\sigma_i = \left[(1/p) \sum_{j=1}^p \| \hat{x}_i - \hat{x}_j \|^2 \right]^{1/2} \quad \dots (2.20)$$

$i = 1..h$

where \hat{x}_j is p nearest centers to \hat{x}_i . In this work value of p is taken equal to 1.

2.4.3 : Calculation of hidden layer output $A_{k,j}$ for for all patterns

Output of the hidden layer ($A_{k,j}$) is given by :

$$A_{k,j} = \exp \left[- \frac{\sum_{i=1}^n (x_{k,i} - \hat{x}_{j,i})^2}{\sigma_j^2} \right] \quad \dots (2.21)$$

$$k = 1 \dots P$$

$$j = 1 \dots h$$

For bias neuron output is set equal to unity.

$$A_{k,(h+1)} = 1$$

$$k = 1 \dots P$$

2.4.4 : Training the weights in second layer

The training of weights in second layer of RBF can be done by simple linear least square regression because output neurons are simple summation units. Therefore the objective is to find the set of weights (w) which minimize the squared norm of the residuals [14,15]:

$$\min_w \| y - w A \|^2 \quad \dots (2.22)$$

where y is a $P \times m$ matrix, A is a $P \times (h+1)$ matrix and w is a $(h+1) \times m$ matrix.

The solution of Eq.(2.22) can be obtained by the pseudo inverse of A and is given by :

$$w = y A^T (A A^T)^{-1} \quad \dots(2.23)$$

where A^T is the transpose of A matrix.

The training of a RBF network is usually an order of magnitude faster than the training of a comparable sized BP network, even with respect to efficient versions of BP. The major limitation of RBF vis a vis BP is that the former requires a much larger number of patterns for training. The pseudo Code for RBF learning in second layer is given below :

Procedure RBF_Learning_in_second_layer

Begin

/* Calculate A matrix */

for i = 1 to P do

begin

for j = 1 to h do

$$A_{i,j} = \exp \left[- \sum_{k=1}^n (x_{i,k} - \hat{x}_{j,k})^2 / \sigma_j^2 \right]$$

$$A_{i,(h+1)} = 1$$

end;

/* Find Weights w */

$$w = T A^T (A A^T)^{-1}$$

End /* End RBF_Learning */

CHAPTER 3

INTEGRATION OF ANN WITH INTELLIGENT PARADIGMS :

In this chapter, Genetic algorithms (GA) and fuzzy logic are integrated with neural networks. Fundamental aspects of genetic algorithms are described in brief in appendix A. Genetic algorithm is implemented essentially as an optimization tool to minimize the error while fuzzy if - then rules are implemented to approximate the functions between inputs and outputs. Sometimes, the knowledge regarding the process is available in form of rules rather than input - output data sets. These rules rather can be mapped on NN using fuzzy systems. The fuzzy logic is useful for representing linguistic uncertainty.

3.1 Backpropagation using Genetic Algorithm

Iterative procedure of backpropagation(BP) is started with a random set of weights. BP works best when solution space is relatively smooth, with few local minima or plateaus. However, if the solution space is convoluted with numerous local minima, then the backpropagation algorithm may get trapped and learning rate may slow down considerably . In such cases it is advised to

restart the training procedure with a new set of different initial random guess for weights. In some cases, however, even after numerous repetitions, BP runs may not yield the optimal solution. GA can be used as an optimization tool to train neural networks [3]. Genetic algorithms do not depend upon the gradients of error to obtain optimal weights and thus it is expected to be more successful in comparison with traditional backpropagation algorithm.

3.1.1 Learning Procedure with GA

The learning procedure with GA consists of the following steps

- (i) Determine representation and calculation of length of chromosome .
- (ii) Generate a random population of chromosomes where each chromosome represents one complete solution (weights) .
- (iii) Decode the chromosome to get real weights .
- (iv) From the weights (from step (iii)) evaluate the network output and the error for each pattern and calculate the total error .
- (v) Calculate fitness for the chromosomes .
- (vi) Repeat Step (iii) to (v) for all chromosomes.
- (vii) Perform selection operation .
- (viii) Cross over the chromosomes .
- (ix) Perform Mutation.
- (x) Save the best solution/chromosome.

(x) Repeat step (iii) to (x) until optimal solution.

Network configuration remains the same as used in backpropagation (in chapter 2). Each of the above steps are described below. The new terms introduced for the genetic algorithm are as follows

length of chromosome for each weight representation = l_w

Total length of chromosome = L_T

Minimum of search space for weights = w_{\min}

Maximum of search space for weights = w_{\max}

Crossover probability = p_c

Mutation probability = p_m

Step (i) : Representation and calculations of length of chromosome

A representation for possible solution must be found before applying GA to a task. The most common way to represent possible solutions is with a bitstring. In this work binary strings of allele are used where each allele is either 0 or 1 . Since architecture of the network is known a priori, a binary string representation containing a fixed number of bits for each weight can be constructed. This representation permits each chromosome to be decoded easily , while still allowing each weight a large degree of freedom.

Calculation for length of each string for a weight

(l_w) is given by

$$2^{l_w} - 1 = \frac{W_{\max} - W_{\min}}{\epsilon_w}$$

or

$$l_w = \log_2 \left[\frac{W_{\max} - W_{\min}}{\epsilon_w} + 1 \right] \quad \dots (3.1)$$

Total length of string (L_T) can be calculated by multiplying l_w with total number of weights.

$$L_T = l_w * \text{total number of weights}$$

or

$$L_T = l_w * \left[\sum_{i=1}^{L-1} (n(i) * n(i+1)) + \text{no of bias weights} \right]$$

or

$$L_T = l_w * \left[\sum_{i=1}^{L-1} (n(i) * n(i+1)) + \sum_{i=2}^L n(i) \right] \quad \dots (3.2)$$

Arrangement of weights in a bitstring

There can be numerous ways of arranging weights in a bitstring. However, the weights in a bitstring should preferably be arranged in a sequence such that those set of weights whose combination directly affects the output are put closer so that the multiple point crossover can not easily destroy such combinations of weights. In this work the weights in a bitstring have been

arranged in the way shown in Fig (3.1)

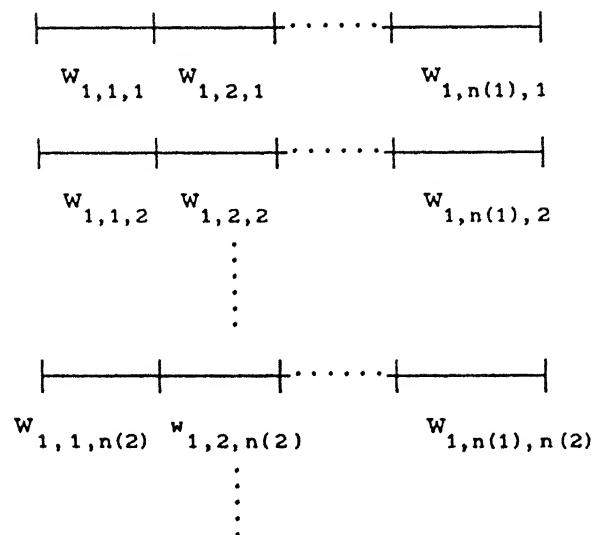


Fig 3.1 : Arrangement of weights in chromosome

It can be seen from Fig (3.1) that weights reaching towards a neuron are put closer rather than weights coming from out a neuron .

Step(ii) : Generation of a new population

Generation of a new population is achieved by creating allele values (either 0 or 1) of each chromosome of population randomly.

Step (iii) : Decoding of chromosome

Each chromosome in the population is to be decoded into the real values. A binary string is first decoded to the digital

value and from the digital value, the actual value of weight can be calculated by the following expression

$$\frac{d * (w_{\max} - w_{\min})}{2^w - 1} + w_{\min} \quad \dots (3.3)$$

where d is the digital value.

Step (iv) : Network output and error

Once the actual weights are known, the network output is calculated (similar to backpropagation). The sum of the squared error is given by

$$E = \sum_{k=1}^P \sum_{j=1}^{n(L)} \left[y_{k,j} - o_{k,L,j} \right]^2 \quad \dots (3.4)$$

Step (v) : Calculate fitness for each chromosome

Now the fitness function is given by the expression

$$\text{fitness} = \left(\frac{1}{1 + E} \right)^\gamma \quad \dots (3.5)$$

where constant γ is generally set equal to one. The factor γ does not have any effect when tournament selection scheme is used for reproduction but it affects when other selection routines (like

roulette wheel selection) are used. Equation 3.5 shows that the lower the error E higher is the fitness ; a perfect network has a fitness of 1. In this work value of γ is taken to be unity.

Step (vi) : Selection operation

Selection operation allows GA to take advantage of a "survival of the fittest" strategy. In this work tournament selection scheme is used. (discussion of Tournament selection is given in appendix A)

Step (vii) : Crossover operation

Crossover (also known as breeding) combines chromosome within the population to produce new chromosome (that did not exist in a previous generation). Multiple point crossover, with crossover probability (p_c) varying from 0.6 to 0.95 has been used in this work.

Step (viii) :Mutation

Simple mutation with probability, p_m , between 0.02 to 0.1 is used .

3.2 GA assisted neural network plus backpropagation combined

BP works best if good initial solution is chosen hence it is advised to use GA to obtain a starting solution for BP which is an approximate solution or close to the optimal solution. In this manner many problems with local minima can be avoided. The major problem with using GA is to decide about the search space for weights. Thus there is the problem of genetic diversity with GA. This problem could be avoided by combining both GA and BP. Following steps need to be added in GA + NN described in section 3.1.1

- (xi) Decode the bestever chromosome found by GA to get real weights.
- (xii) Start backpropagation with these weights ; decide about the learning rate and momentum factor.
- (xiii) Do forward processing (evaluate the network output) for a given data set .
- (xiv) Do backward processing (change weights by calculating gradients etc.) for that data set .
- (xv) Calculate error in that data set.
- (xvi) Repeat step (xiii) to (xv) for all data sets.
- (xvii) Repeat from step (xiii) to step (xvi) until termination criterion is met with . In the present work termination criteria is fixed as number of maximum iterations/generations.

3.3 Fuzzy Neural Network

There are numerous methods to deal with uncertainty in

expert systems [4]. Fuzzy logic is one of such approach in which the uncertainty is represented by possibility distributions for the antecedents and the consequent of the rule. An uncertain input can be propagated to the consequent by a variety of mechanisms. The utility of the fuzzy sets lies in their capability to model uncertain or ambiguous data often encountered in real process. There have been various attempts to make a fusion of fuzzy logic and neural network so as to improve the performance in a decision making system [16].

A fuzzy layered neural network for classification and rule generation can be created by using logical neurons. It can handle uncertainty and/or impreciseness in the input as well as in the output. Logical AND/OR operation can be done in place of weighted sum/product and sigmoid function, as in the case of backpropagation.

ANFIS : Adaptive Network Based Fuzzy inference System [5]

Adaptive network based fuzzy inference system (ANFIS) is a fuzzy inference system which has been implemented in the framework of adaptive networks. By using a hybrid learning procedure, ANFIS can use human knowledge to construct an input - output mapping based on fuzzy if-then rules and stipulated input output data pairs. In the present work ANFIS architecture is employed to model nonlinear functions /mappings. Before discussing about the ANFIS architecture it is necessary to understand the fuzzy if then rules and fuzzy inference system itself.

3.3.1 Fuzzy if- then rules and fuzzy inference system

(i) Fuzzy if then rules

Fuzzy if -then rules or fuzzy conditional statements are expressions of the form '**IF A then B**' , where A and B are labels of fuzzy sets, characterized by appropriate mapping functions. The if then type of rules are used to capture the process knowledge. This knowledge is generally expressed by operators and hence it has built in linguistic uncertainty. An simple example is

if Oxygen Supplied is more, then Carbon content of Steel is low

where *Oxygen Supplied* and *Carbon content of steel* are the linguistic variables, and *more* and *low* are linguistic values or labels that are characterized by membership functions.

Through the use of linguistic labels and membership functions, the fuzzy if - then rule can easily capture the spirit of a " rule of thumb " used for a process, from the data examples provides to it.

(ii) Fuzzy Inference System

Fuzzy inference systems are also known as fuzzy rule based systems, fuzzy models, fuzzy associative memories (FAM) , or fuzzy controllers (when used as controllers). Basically a fuzzy function unit is composed of five functional blocks (see Fig

(3.2))

- . a fuzzification interface which transforms the crisp inputs in to degree of match with linguistic values;
- . a rule base containing a number of fuzzy if then rules ;
- . a data base which defines the membership functions of the fuzzy sets used in the fuzzy rules ;
- . a decision making unit which performs the interface options on the rules ;
- . a defuzzification interface which transforms the fuzzy results of the inference in to the crisp output.

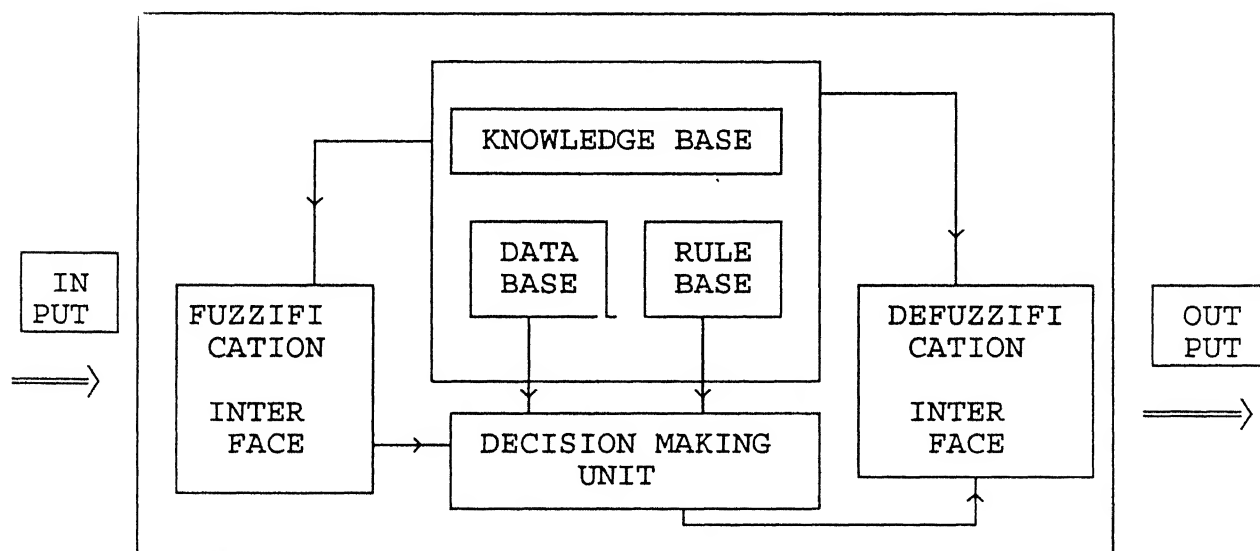


Fig 3.2 : Fuzzy Inference System [5]

Usually, the rule base and the data base are jointly referred to as knowledge base.

The sequence of steps of the fuzzy reasoning (inference operation upon fuzzy if - then rules) performed by fuzzy inference systems are

Step 1 : Fuzzification

Compare the input values with the membership functions on the premise part to obtain the membership values (or compatible measures) of each linguistic labels.

Step 2 : Fire the rules

Combine through a T - norm operator (usually multiplication or min) the membership function values on the premise part to get firing strength (weight) of each rule.

Step 3: Generate the qualified rule

Generate the qualified consequent (either fuzzy or crisp) of each rule depending upon its firing strength.

Step 4 : Defuzzification

Aggregate the qualified consequent to produce a crisp output.

In a fuzzy inference system the following fuzzy reasoning are used in the present work. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output.

CENTRAL LIBRARY
 I. I. T., KANPUR
 12/10/98
 Acc. No. A.

3.3.2 · Architecture of Generalized Adaptive Network and learning algorithm

Adaptive network is a superset of all kinds of fuzzy feed forward neural networks with supervised learning capability. An adaptive network, as the name implies, is a network structure consisting of nodes and directional links through which the nodes are connected. Moreover, a part or all of the nodes are adaptive, which means their output depends on the parameter(s) pertaining to these nodes, and the learning rule specifies how these parameters should be changed to minimize a prescribed error measure.

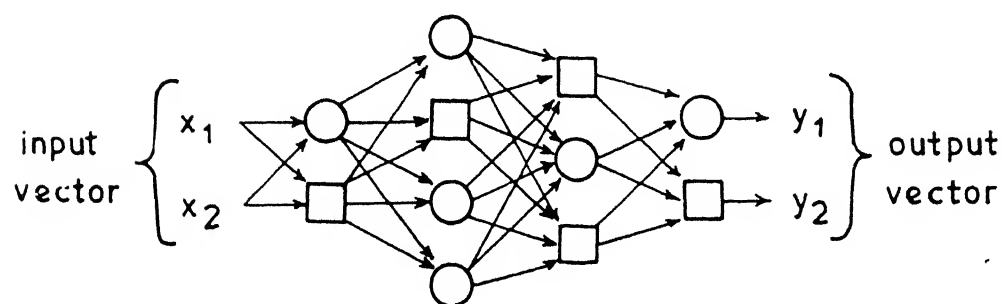
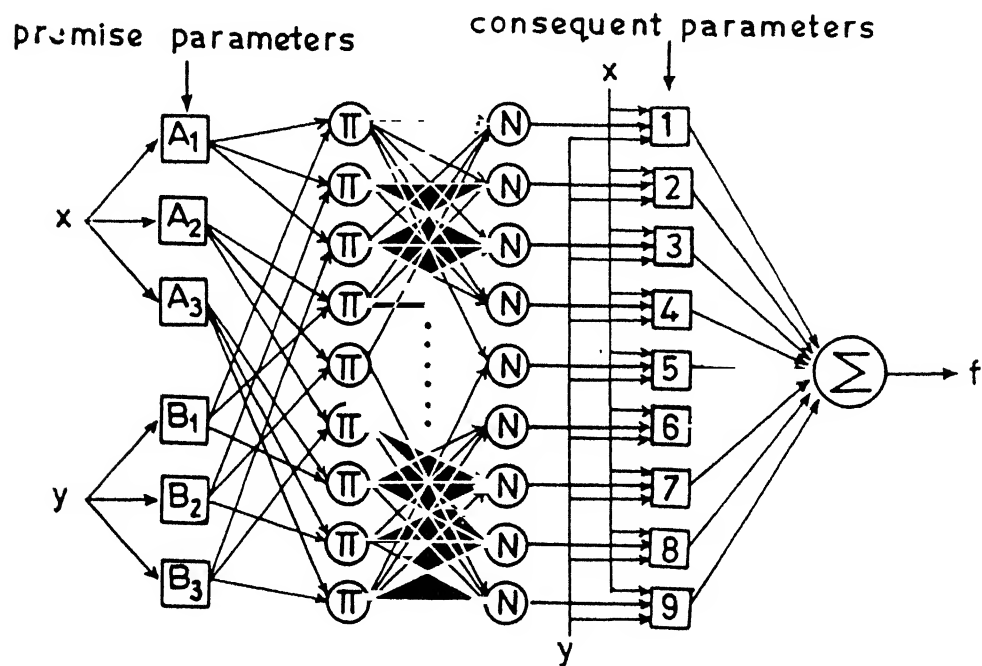
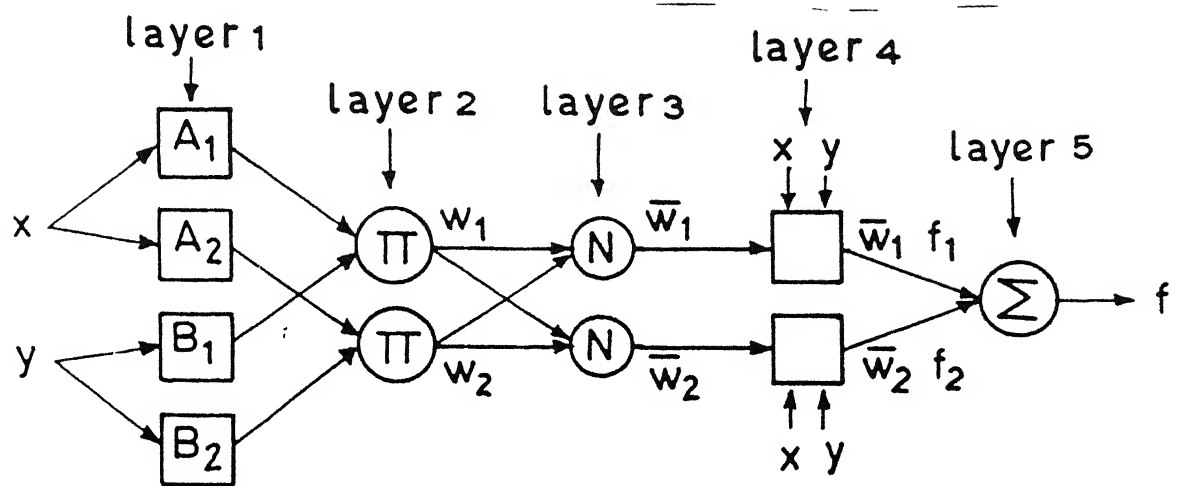


Fig (3.3) : Adaptive Network



EQUIVALENT FNN FOR TWO INPUTS AND NINE RULES

Fig 3.4 a,b : ANFIS Networks

An adaptive Network (Fig (3.3)) is essentially a multi layer feed forward network in which each node performs a particular function (node function) on incoming signals as well as a set of parameters specified for each node. The formula for the node functions may vary from node to node, the choice of each node function vary from node to node, and the choice of the node functions depends on the overall input - output function which the adaptive network is required to carry out . The links in the adaptive network only indicate the flow direction of the signals between nodes . No weights are associated with the links. To reflect different adaptive capabilities, both circle and square nodes are used in the adaptive network. A square node (adaptive node) has parameters while a circle node (fixed node) has none . The parameter set of an adaptive network is the union of the parameter sets of each adaptive node. In order to achieve a desired input - output mapping these parameters are updated according to the given training data and a gradient based learning procedure . Gradient based methods are notorious for their slowness and they usually get trapped in local minima. In ANFIS hybrid learning rule is used to speed up the learning process. The batch type learning is used in this work. The convention used to describe the various elements are mentioned below.

Number of inputs = n

Number of outputs = m

Number of rules for i^{th} input outputs = r_i
 $i = 1 \dots n$

Number of patterns = P

Input matrix = $x_{k,j}$
 $k = 1 \dots p$
 $j = 1 \dots n$

Output matrix = $y_{k,j}$
 $k = 1 \dots P$
 $j = 1 \dots m$

Each input i has r_i number of rules. For k^{th} data set, for each input i, output of each rule is given by $\mu_{k,i,j}$ which is a function of $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$.

Output of different rules = $\mu_{k,i,j}$
 $k = 1 \dots P$
 $i = 1 \dots n$
 $j = 1 \dots r_i$

Number of rules fired = h

For k^{th} pattern finding strength for each rule = $w_{k,j}$
 $k = 1 \dots p$
 $j = 1 \dots h$

Consequent parameter = $p_{i,j,l}$

$i = 1 \dots m$

$j = 1 \dots n+1$

$l = 1 \dots h$

Normalized final strength is given by = $\bar{w}_{k,j}$

$k = 1 \dots P$

$j = 1 \dots h$

The algorithm utilizing the above element is now described.

Layer 1

In layer 1 all the nodes are square nodes (Fig (3.4 a,b)). Each node has three parameters a, b and c , are called as the premise parameters. In layer 1 fuzzification of inputs is done. Each of the input i is associated with r_i number of square nodes for fuzzification, thus r_i is the number of Membership Function's (MF) associated with the i^{th} input. The number of MF's for each input (r_i) represents the number of linguistic labels for i^{th} input. For example, one input can be associated with 3 MFs namely small, medium and large. The output of each of the node (fuzzified value) in this layer is calculated by bell shaped pi function given below.

$$\mu(x, a, b, c) = \frac{1}{1 + \left[\left(\frac{x - c}{a} \right)^2 \right]^b} \quad \dots (3.6)$$

where x is the input value to the node .

The learning algorithm has to find out the appropriate values of a, b, c by an iterative procedure. Procedure for calculation of initial values of a, b, c is discussed in section 3.2.2 (a). In network of type Fig (3.4 a) the imposed condition is that each input is associated with equal number of square nodes MF's represented by square node (say r). Thus

$$r_i = r \quad \forall i = 1 \dots n \quad \dots (3.7)$$

The iterative procedure for the adjustment of premise parameters with iteration's is discussed in section 3.2.2 c.

Layer 2

Each node in layer 2 is a circle node labeled π (Fig (3.4 a,b)) designed for "and" operation on the incoming signals. The number of nodes in the second layer represents the total number of rules fired by the network. The output of "and" operation is called *firing strength* of that particular rule. In this work simple multiplication has been used as an "and" operation.

For the network of type Fig (3.4 a), the number of nodes in the second layer (h) is equal to the number of MF associated with each input (r) [$h = r = r_i$] The firing strength of each rule is calculated by simple multiplication of the incoming signals.

$$w_{k,j} = \prod_{i=1}^n \mu_{k,i,j} \quad \dots (3.8)$$

$k = 1 \dots P$ (for each pattern)

$j = 1 \dots h$ (for each node)

For network of type Fig (3.4 b), the number of rules (h) is given by

$$h = \prod_{i=1}^n r_i \quad \dots (3.9)$$

For example, if the number of inputs is 4 and each input is associated with 2 MF's then the number of rules for this type of network will be 2^4 (= 16). The firing strength for each rule for this type of network (Fig (3.4 b)) is calculated by

$$w_{k,j} = \prod_{i=1}^n \mu_{k,i} S_{j,i} \quad \dots (3.10)$$

where S matrix is given by

$$\begin{aligned}
S_1 &= \{ 1, 1, 1, \dots, 1 \} \\
S_2 &= \{ 1, 1, 1, \dots, 2 \} \\
&\vdots \\
S_r &= \{ 1, 1, 1, \dots, r_n \} \\
&^n = \{ 1, 1, 1, \dots, 2, 1 \} \\
&= \{ 1, 1, 1, \dots, 2, 2 \} \\
&\vdots \\
&= \{ 1, 1, 1, \dots, 2, r_n \} \\
&\vdots
\end{aligned}$$

.. (3.11)

Layer 3

Each node in layer 3 is a circular node and is depicted as by N in Fig (3.4 a.b) where the symbol N refers to the normalized strength of each rule. Total number of nodes in this layer is same as that in the previous layer (i.e. h) . Normalized firing strength is given by :

$$\bar{w}_{k,j} = \frac{w_{k,j}}{\sum_{j=1}^n w_{k,j}} \quad \text{.. (3.12)}$$

Layer 4

Each node in layer 4 is a square node. Parameters in this layer are known as consequent parameters, p. The number of nodes in layer 4 is m*h where m is the total number of output and h is the nodes in 2nd and 3 rd layer .In case of one output (as in two examples of dephosphorization and desulphurization) the

number of nodes in this layer is same as in the second and third layer (i.e. h nodes) . The number of parameters associated with each node are $n+1$,where n is the total number of inputs. Thus the total number of consequent parameters of the network will be $(n+1) * h * m$.Each node receives the normalized firing strength from the previous layer (third layer). In addition to the normalized firing strength of the rule each node in this layer is supplied once again with input patterns Let f be defined as :

$$f_{k,i,l} = \left(\sum_{j=1}^n x_{k,j} * p_{i,j,l} \right) + p_{i,n+1,l} \quad \dots (3.13)$$

$$k = 1..P \text{ (number of patterns) }$$

$$i = 1..m$$

$$l = 1..h$$

where x is input matrix and p 's are the consequent parameter.

Output of each node , i , in the fourth layer is the product of f and the normalized firing strength (from previous layer) . Let $O_{i,j}^4$ be output of a particular node in the fourth layer then for the k^{th} pattern it's value is calculated from :

$$O_{i,j}^4 = \bar{w}_i * f_{k,j,i} \quad \dots (3.14)$$

$$k = 1..P$$

$$i = 1..h$$

$$j = 1..m$$

The learning algorithm is designed to find appropriate values of consequent parameter (p 's) by an iterative procedure.

The way to calculate the consequent parameters is discussed in section 3.3.2 (b).

Layer 5

Each node in the layer 5 is a circular node and the number of nodes is equal to number of outputs m . The output $(O_{k,i})$ of the nodes in this layer is the final output of the network and is the sum of the output of the h nodes in the fourth layer.

$$\begin{aligned}
 O_{k,i} &= \sum_{l=1}^h f_{k,i,l} * \bar{w}_{k,l} \\
 &= \frac{(\sum_{l=1}^h f_{k,i,l} * w_{k,l})}{\sum_{l=1}^h w_{k,l}} \quad \dots (3.15)
 \end{aligned}$$

$k = 1 \dots P$
 $i = 1 \dots m$

3.3.2 (a) Initial guess for premise parameters in layer 1

Initial guess values are chosen in such a way that for each value of input, at least one of the membership value is greater than 0.5.

of $(n + 1) \times m$ size and the matrix y which is the also target matrix, is of $P \times m$ size.

The consequent parameters are modified only once in each epoch, as in the batch type learning. Thus, the consequent parameters are adjusted to minimize the norm of error in each epoch. The norm of error is given by :

$$E = \min_x || A^*X - Y || \quad \dots (3.20)$$

The next step is to adjust the premise parameters.

3.3.2 c : Adjustment of premise parameters in layer 1

Premise parameters a, b, c of the bell shaped MF's in layer 1 are also adjusted once in each epoch in contrast with BP. The change in premise parameters is calculated from :

$$(\Delta a_{i,j}) = - \eta \sum_{k=1}^P \left(\frac{\partial E_k}{\partial a_{i,j}} \right) \quad \dots (3.21)$$

$$(\Delta b_{i,j}) = - \eta \sum_{k=1}^P \left(\frac{\partial E_k}{\partial b_{i,j}} \right) \quad \dots (3.22)$$

$$(\Delta c_{i,j}) = -\eta \sum_{k=1}^P \left(\frac{\partial E_k}{\partial c_{i,j}} \right) \quad \dots (3.23)$$

$$i = 1 \dots n$$

$$j = 1 \dots r_i$$

where η is called learning rate and is calculated by the equation :

$$\eta = \frac{S}{\sum_{i=1}^n \sum_{j=1}^{r_i} \left[\left[\frac{\partial E_k}{\partial a_{i,j}} \right]^2 + \left[\frac{\partial E_k}{\partial b_{i,j}} \right]^2 + \left[\frac{\partial E_k}{\partial c_{i,j}} \right]^2 \right]} \quad \dots (3.24)$$

where S is the step size and initially chosen between 0 to 1. The value of S is modified in each epoch . The description of the modification step size S is given in section 3.3.2 (d) .

The derivatives of error with respect to premise parameters (a,b,c) $\{ \partial E_k / \partial a, \partial E_k / \partial b, \partial E_k / \partial c \}$ mentioned in Eqns.(3.21 -3.24) are calculated by the learning algorithm and is given by Eqns.(3.25-3.27) :

$$\frac{\partial E_k}{\partial a_{i,j}} = \frac{\partial E_k}{\partial \mu_{k,i,j}} * \frac{\partial \mu_{k,i,j}}{\partial a_{i,j}} \quad \dots (3.25)$$

$$\frac{\partial E_k}{\partial b_{i,j}} = \frac{\partial E_k}{\partial \mu_{k,i,j}} * \frac{\partial \mu_{k,i,j}}{\partial b_{i,j}} \quad \dots (3.26)$$

$$\frac{\partial E_k}{\partial c_{i,j}} = \frac{\partial E_k}{\partial \mu_{k,i,j}} * \frac{\partial \mu_{k,i,j}}{\partial c_{i,j}} \quad \dots (3.27)$$

$$i = 1 \dots n$$

$$j = 1 \dots r_i$$

Derivatives of membership function with premise parameters $\{ \partial \mu / \partial a, \partial \mu / \partial b, \partial \mu / \partial c \}$ are calculated by differentiating the Eq.(3.6) with a,b and c and are given by :

$$\frac{\partial \mu}{\partial a} = \frac{2b}{a} * \mu(1-\mu) \quad \dots (3.28)$$

$$\frac{\partial \mu}{\partial b} = \mu(1-\mu) * \log \left[\left((x-c)/a \right)^2 \right] \quad \dots (3.29)$$

$$\frac{\partial \mu}{\partial c} = \frac{2b}{x-c} * \mu(1-\mu) \quad \dots (3.30)$$

Error derivative with respect to membership values are calculated by following chain equations :

$$\frac{\partial E_k}{\partial \mu_{k,i,j}} = \sum_{l=1}^h \left[\frac{\partial E_k}{\partial w_{k,l}} * \frac{\partial w_{k,l}}{\partial \mu_{k,i,j}} \right] \quad \dots (3.31)$$

$$\frac{\partial E_k}{\partial w_{k,l}} = \sum_{i=1}^m \left[\frac{\partial E_k}{\partial O_{k,i}} * \frac{\partial O_{k,i}}{\partial w_{k,l}} \right] \quad \dots (3.32)$$

$$\frac{\partial O_{k,i}}{\partial w_{k,l}} = \frac{[-O_{k,i} + f_{k,i,l}]}{\sum_{j=1}^h w_{k,j}} \quad \dots (3.33)$$

$$\begin{aligned} k &= 1 \dots P \\ l &= 1 \dots h \\ i &= 1 \dots m \end{aligned}$$

AND

$$\frac{\partial E}{\partial O_{k,j}} = -2 * (Y_{k,j} - O_{k,j}) \quad \dots (3.34)$$

$$\begin{aligned} k &= 1 \dots P \\ j &= 1 \dots m \end{aligned}$$

3.3.2 d : Modification of Step Size S

Step size S is modified after 4 epochs according to two linguistic rules :-

1. If error continues to decrease during last four epochs then increase S by 10 % .
2. If the error undergoes fluctuations (2 times increase and 2 times decrease) then reduce S by 10 % .

3.4 GA + ANFIS

In GA + ANFIS [17] the error in prediction is minimize with the Genetic algorithms. Premise parameters (a,b,c in the first layer) are adjusted by GA's while consecutive parameters are calculated inverse method as ANFIS described in section 3.3.2 (b). One important point while implementing GA in ANFIS is to decide about the search space of premise parameters a,b,and c. The details are given below :

Search space for "a" parameter of MF :

The range in which optimum value of the premise parameter " $a_{i,j}$ " is searched , is $[0, (\max_i - \min_i)]$ where \max_i and \min_i are the maximum and minimum values, respectively, of the i^{th} input.

Search space for "b" parameter of MF :

The range in which optimum value of the premise parameter " $b_{i,j}$ " is searched , is $[2,3]$. On putting $\varepsilon = 0.05$ in Eq.2.17 value of $b_{i,j}$ comes out 2.124., therefore search space is kept between $[2,3]$.

Search space for "c" parameter of MF :

The range in which optimum value of the premise parameter " $a_{i,j}$ " is searched , is $[\min_i, \max_i]$.

Binary coding has been used. Premise parameters a,b and c are grouped in the binary string. Binary tournament selection is used as a selection scheme which is described in appendix A. Multiple point cross over is used with 4 number of crossover sites. Simple mutation with probability 0.05 is used. Both types of network (Fig. (3.4 a,b)) are trained with this algorithm.

CHAPTER 4

RESULTS AND DISCUSSION

The algorithms described in chapters 2 and 3 are applied to two problems of dephosphorization and desulphurization in iron and steel making. Description of the problem and technical details of data set are given in chapter 1. The results of backpropagation and it's variation are discussed in section 4.1. Results of RBF are presented in section 4.2 and results of GA + NN are discussed in section 4.3. Results of ANFIS on both the problems are discussed in section 4.4 and results of GA + ANFIS are given in section 4.5. A critical comparison of various algorithms developed in this work is done in section 4.6 .

The terminology used for discussing the results of various algorithms remains the same as discussed in chapters 2 and 3. However, some of the new terms introduced in this chapter are given below :

Final learning error = E_F

Correlation coefficient obtained during testing
of seen data = r_L

Correlation coefficient obtained during testing
of unseen data = r_T .

Data scaling in BP (and its four variations), RBF and GA+BP, is done in the range of $[\min_i, \max_i]$ for each (i^{th}) input (Eq.(2.4)). These terms are also used in ANFIS (Eqns.(3.16-3.18)). The values of all the terms used for the problems of dephosphorization and desulphurization are shown in Table 4.1,4.2 respectively.

4.1 : Results of backpropagation and its variations

Theoretical background of backpropagation has been discussed in section 2.1 and it's variations are discussed in section 2.3. The results of BP and it's variation on the two problems are given in this section.

4.1.1 : Dephosphorization problem

Backpropagation and its four variations are run for 10,000 epochs, for 7 network topologies. In all the cases, the

starting weights are generated between small ranges, typically chosen between -0.5 to 0.5 . Both training and testing data are scaled between 0.1 to 0.9 (Eq.(2.4)). Learning rate and momentum factor are set equal to 0.8 to 0.5, respectively, (except in variation 4 where both learning rate and momentum factor are kept very small).

Tables 4.3 to 4.7 present the results of backpropagation and its variations for 7 different topologies. It can be deduced from Tables 4.3 to 4.7 that on increasing the number of nodes in hidden layers, the training error reduces. From the prediction results it can be concluded that for bigger networks, the results are good for training patterns but poor for testing(i.e. poor generalization); for example in Table 4.3 r_L varies between 0.8 to 0.93 during learning whereas r_T varies between 0.24 to 0.69 during testing.

If patterns are presented in random order in each epoch (variation 1) the generalization capability of network improves. (for the 7-6-1 network r_L and r_T for BP are 0.832 and 0.539, respectively (Table 4.3), whereas r_L and r_T for variation 1 are 0.831 and 0.806 (Table 4.4)), respectively.

According to variation 2 (discussed in section 2.3.2) patterns are presented in proportion to their error values at each epoch. When this is done the error does not decrease as rapidly as expected and learning also slows down. Final error E_F in BP for 7-6-1 network is 0.076 (Table 4.3) and for variation 2, final

error (E_F) is 0.128 (Table 4.5).

The logarithmic error function (variation 3 discussed in section 2.3.3) has been used to speed up the convergence [12]. However, similar to the variation 2, this variation also does not improve the convergence of BP. As seen from Table 2.3 and 2.7, the variation 3 converges to higher error values than BP for all the 7 topologies tried. For example for 7-6-1 network in BP, the error E_F is 0.076 (Table 4.3) and for variation 3, the error E_F is 0.082 (Table 4.6).

Logarithmoid function (variation 4 in section 2.3.4) is used as an alternative to sigmoidal function. It was claimed [13] that there is no need of data preprocessing when using logarithmoid function. Therefore, both preprocessed and raw data are used to train the network. The value of learning rate and momentum factor are taken very small, typically chosen as equal to 0.01.

With scaled data (scaled between 0.1 to 0.9, similar to BP and the variations 1 to 3) the value of c in logarithmoid function (Eq.(2.17)) is typically taken as 1. Results of variation 4 with preprocessed data are shown in Table 4.7. The variation 4 gives better results than BP when input -output scaling is done (for example for 7-10-1 network r_L and r_T for BP are 0.801 and 0.580, respectively (Table 4.3), whereas r_L and r_T for variation 4 are 0.794 and 0.859 respectively (Table 4.7)). In variation 4, amongst all the 7 topologies tried, the results do not vary much.

(for example r_L varies from 0.788 to 0.811 and r_T varies from 0.850 to 0.862 (Table 2.7)).

The objective of using logarithmoid function is that no data preprocessing is needed. The network is therefore, run with raw data. The results are shown in Table 2.8. Error is shown on the scaled basis. When network is trained with raw data, logarithmoid function gives very poor results (for example for 7-6-1 network in BP, error E_F is 0.076 (Table 4.3) and for variation 4 with raw data error E_F is 0.137 (with $c = 1$) (Table 4.8)). In this work training of network with different c values were also tried out and results are presented in Table 4.8.

From the results presented in Table 4.3-4.8 it can be seen that, amongst all the variations tried, the results of variation 1 (i.e. when patterns presented in random order in each epoch) are the best compared to all the variations of BP; the training time of this variation is almost the same as that of BP.

4.1.2 : Desulphurization problem

Similar to the case of dephosphorization problem, 7 different network topologies are tried. The results of BP and its variations on desulphurization problem are given in Tables 4.9-4.13. Scaling of data is done in same way as for dephosphorization problem. Learning rate, momentum factor and initial random weight generation ranges are also similar to the

case of dephosphorization problem i.e. 0.8,0.5 and [-0.5,0.5], respectively.

It can be seen that variation 1 (Table 4.10) gives a better performance than BP (Table 4.9), and variation 2 (Table 4.11) and variation 3 (Table 4.12) give poor results. With scaled data, variation 4 (Table 4.13) gives better results than BP. The trend of these results matches with dephosphorization problem.

4.2 Results of Radial Basis Function

Results of RBF are shown in Table 4.14.4.15, for different network topologies. Number of nodes in the middle layer represent the number of centers and this number was varied in different networks. On training the first layer by K- mean algorithm, initial centers are chosen randomly from the training data. Convergence of K-mean is achieved when the change in clustering centers is less than some specified low value, typically taken to be equal to 0.001.

4.1.1 :An example of RBF learning algorithm on Dephosphorization problem

Ten patterns of dephosphorization problem taken from the actual data set (presented in section 1.2) are used to explain the actual working of the algorithm. These patterns are given below

| Hot metal Weight (in tons) | T in °C | C in % x 100 | Mn in % x 100 | Initial P in % x 100 | Ore (Kg) | Total O ² Blown (Nm ³) | Final P % x 100 |
|----------------------------------|---------------|-----------------------|------------------------|-------------------------------|-------------|--|--------------------------|
|----------------------------------|---------------|-----------------------|------------------------|-------------------------------|-------------|--|--------------------------|

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 315.0 | 1613 | 447 | 314 | 26 | 1000 | 14100 | 11 |
| 316.1 | 1606 | 384 | 330 | 23 | 900 | 13860 | 11 |
| 316.1 | 1620 | 223 | 312 | 22 | 500 | 13220 | 14 |
| 315.5 | 1606 | 543 | 349 | 27 | 1044 | 14240 | 14 |
| 311.0 | 1611 | 496 | 333 | 25 | 500 | 15020 | 12 |
| 314.3 | 1636 | 257 | 314 | 30 | 1000 | 14040 | 16 |
| 314.9 | 1642 | 425 | 338 | 32 | 1064 | 14500 | 16 |
| 316.2 | 1642 | 332 | 334 | 25 | 1020 | 13940 | 14 |
| 314.3 | 1616 | 371 | 306 | 38 | 1020 | 14150 | 16 |
| 313.5 | 1602 | 529 | 303 | 22 | 880 | 14850 | 9 |

Initially this data set is scaled between 0.1 to 0.9 (Eq.(2.4)). The minimum and maximum values for the data scaling are taken from the data itself presented above. After scaling the transformed patterns are presented below.

Pattern No : 1
0.7154 0.3200 0.6600 0.2913 0.3000 0.8092 0.4911 0.3286

Pattern No : 2
0.8846 0.1800 0.5025 0.5696 0.1500 0.6674 0.3844 0.3286

Pattern No : 3
0.8846 0.4600 0.1000 0.2565 0.1000 0.1000 0.1000 0.6714

Pattern No : 4
0.7923 0.1800 0.9000 0.9000 0.3500 0.8716 0.5533 0.6714

Pattern No : 5
0.1000 0.2800 0.7825 0.6217 0.2500 0.1000 0.9000 0.4429

Pattern No : 6
0.6077 0.7800 0.1850 0.2913 0.5000 0.8092 0.4644 0.9000

Pattern No : 7
0.7000 0.9000 0.6050 0.7087 0.6000 0.9000 0.6689 0.9000

Pattern No : 8
0.9000 0.9000 0.3725 0.6391 0.2500 0.8376 0.4200 0.6714

Pattern No : 9

0.6077 0.3800 0.4700 0.1522 0.9000 0.8376 0.5133 0.9000

Pattern No : 10

0.4846 0.1000 0.8650 0.1000 0.1000 0.6390 0.8244 0.1000

The terminology used and the algorithm are described in chapter 2 (section 2.4). Initially, suppose 3 centers are chosen by the user. Input layer has 7 nodes (variables), and there are 3 clustering centers in hidden layer (plus one bias node) and one node (variable) in output layer. Thus, the network configuration will be 7-4-1. The bias node in middle layer always has unit output. Algorithm can be divided in four steps :

- (i) Calculate the clustering center from K-mean clustering algorithm.
- (ii) Calculate the width of the centers.
- (iii) Calculate the hidden layer output for all patterns (A matrix).
- (iv) Calculate weight(w) between hidden and output layer.

Now the calculations in these four steps are given below in detail.

Step (i) K_mean clustering algorithm

(a) Initialize the center

Three initial centers are chosen randomly from the data set used for learning. Three random numbers generated are 5,3 and 10.

Center Number 1 is the pattern Number 5 :

0.100000 0.280000 0.782500 0.621739 0.250000 0.100000 0.900000

Center Number 2 is the pattern Number 3 :

0.884614 0.460000 0.100000 0.256522 0.100000 0.100000 0.100000

Center Number 3 is the pattern Number 10 :

0.484614 0.100000 0.865000 0.100000 0.100000 0.639007 0.824444

(b) Calculate the new centers on the basis of euclidean distance basis until there is any change in centers.

Euclidean distance is defined as $\sum_{i=1}^n (z_{i,1} - z_{i,2})^2$ between two points $z_{i,1}$ and $z_{i,2}$ of dimension n (in the example shown $n=7$). The distance of different patterns with each center is calculated to find out which pattern is the closest to which center.

(i) Iteration 1

| Pattern Number | Distances from centers | | | Closest center from pattern |
|-------------------|------------------------|----------|----------|--------------------------------|
| | 1 | 2 | 3 | |
| 1 | 1.084976 | 1.029082 | 0.600300 | 3 |
| 2 | 1.142127 | 0.862396 | 0.845696 | 3 |
| 3 | 1.381923 | 0.000000 | 1.309455 | 2 |
| 4 | 1.142854 | 1.415772 | 0.965616 | 3 |
| 5 | 0.000000 | 1.381923 | 0.882090 | 1 |
| 6 | 1.314969 | 0.991632 | 1.138032 | 2 |
| 7 | 1.264665 | 1.378697 | 1.211121 | 3 |
| 8 | 1.402583 | 1.040874 | 1.253504 | 2 |
| 9 | 1.304586 | 1.259269 | 1.014159 | 3 |
| 10 | 0.882090 | 1.309455 | 0.000000 | 3 |

Now the new centers are calculated by taking the mean of the patterns which are closer to that particular center in previous iteration.

New Center Number 1 is now mean of 5th pattern only. The new center number 1 is thus given by :

0.100000 0.280000 0.782500 0.621739 0.250000 0.100000 0.900000

New Center Number 2 is now mean of 3rd and 6th and 8th patterns. The new center number 2 is thus given by

0.797435 0.713333 0.219167 0.395652 0.283333 0.582270 0.328148

New Center Number 3 is now mean of 1,2,4,7,9 and 10th patterns. The new center number 3 is thus given by

0.697434 0.343333 0.667083 0.453623 0.400000 0.787470 0.572593

These centers are different than initial centers which were chosen randomly at start. Therefore, convergence has not been achieved and more iterations are needed until the difference of centers between two consecutive steps is less than 0.001 (say).

(ii) Iteration 2

| Pattern Number | Distances from centers | | | Closest center from pattern |
|-------------------|------------------------|----------|----------|--------------------------------|
| | 1 | 2 | 3 | |
| 1 | 1.084976 | 0.667089 | 0.210657 | 3 |
| 2 | 1.142127 | 0.656322 | 0.463214 | 3 |
| 3 | 1.381923 | 0.650832 | 1.093179 | 2 |
| 4 | 1.142854 | 1.068295 | 0.546932 | 3 |
| 5 | 0.000000 | 1.266239 | 1.002414 | 1 |
| 6 | 1.314969 | 0.411742 | 0.692570 | 2 |
| 7 | 1.264665 | 0.780088 | 0.663880 | 3 |

| | | | | |
|----|----------|----------|----------|---|
| 8 | 1.402583 | 0.450436 | 0.721387 | 2 |
| 9 | 1.304586 | 0.865490 | 0.628598 | 3 |
| 10 | 0.882090 | 1.123248 | 0.666479 | 3 |

New Center Number 1 is now mean of 5th pattern only.

The new center number 1 is thus given by

0.100000 0.280000 0.782500 0.621739 0.250000 0.100000 0.900000

New Center Number 2 is now mean of 3rd and 6th and 8th patterns. The new center number 2 is thus given by :

0.797435 0.713333 0.219167 0.395652 0.283333 0.582270 0.328148

New Center Number 3 is now mean of 1,2,4,7,9 and 10th patterns. The new center number 3 is thus given by

0.697434 0.343333 0.667083 0.453623 0.400000 0.787470 0.572593

These new centers are similar to that of previous iteration (within the specified), therefore convergence has been achieved. The three centers are

0.100000 0.280000 0.782500 0.621739 0.250000 0.100000 0.900000
 0.797435 0.713333 0.219167 0.395652 0.283333 0.582270 0.328148
 0.697434 0.343333 0.667083 0.453623 0.400000 0.787470 0.572593

This is the end of K-mean algorithm.

Step (ii) : Calculation of RBF width

| Centers Number | Distance from other centers | | | Distance from the other closest center (d) | RBF width of the center (p =1) |
|-------------------|--------------------------------|----------|----------|---|---|
| | 1 | 2 | 3 | | |
| 1 | - | 1.266239 | 1.002414 | 1.002414 | 1.002414 |
| 2 | 1.266239 | - | 0.682907 | 0.682907 | 0.682907 |
| 3 | 1.002414 | 0.682907 | - | 0.682907 | 0.682907 |

The value of d in each row is minimum of all the values in columns 1,2,3. The important point to note in calculating distance from the centers is not to consider self distances.

Step (iii) : Calculation of hidden layer output (A matrix)

Hidden layer output (A matrix) is calculated from Eq.(2.21) and is given below. The output of bias node is set equal to unity. Therefore, elements of the 4th column in matrix A are 1.0 for all patterns.

$$A = \begin{bmatrix} 0.309899 & 0.385116 & 0.909232 & 1.000000 \\ 0.273028 & 0.397064 & 0.631229 & 1.000000 \\ 0.149490 & 0.403221 & 0.077114 & 1.000000 \\ 0.272578 & 0.086540 & 0.526543 & 1.000000 \\ 1.000000 & 0.032128 & 0.115947 & 1.000000 \\ 0.178919 & 0.695226 & 0.357543 & 1.000000 \\ 0.203583 & 0.271211 & 0.388659 & 1.000000 \\ 0.141172 & 0.647230 & 0.327630 & 1.000000 \\ 0.183828 & 0.200648 & 0.428582 & 1.000000 \\ 0.461008 & 0.066845 & 0.385788 & 1.000000 \end{bmatrix}$$

Step (iv) Calculation of second layer connection weight(w)

The weights in second layer are calculated by taking pseudo inverse of A matrix (Eq.(2.23)) .Target vector y is shown below

$$y = \begin{bmatrix} 0.328571 \\ 0.328571 \\ 0.671429 \\ 0.671420 \\ 0.442857 \\ 0.900000 \\ 0.900000 \\ 0.671429 \\ 0.900000 \\ 0.100000 \end{bmatrix}$$

Finally the weight vector (between four nodes (3 centers + 1 bias node) and one output node) calculated from the pseudo inverse of A (Eq.(2.23)) is shown below

$$w = \begin{bmatrix} -0.606239 \\ 0.057936 \\ -0.509943 \\ 0.976903 \end{bmatrix}$$

The above weights in combination with clustering centers and RBF width can now be used to calculate the output for an unseen pattern.

4.2.2 : Results of dephosphorization problem

It can be deduced from Table 4.14 that as the number of hidden nodes are increased, the learning capability of RBF increases. The network 7-60-1 shows the best generalized results as compared to all other topologies ($r_L = 0.820320$, $r_T = 0.785239$). Increasing the number of nodes in hidden layer improves the performance during training but the generalization ability is adversely affected (for example, for 99 centers

(7-100-1 network) $E_F = 0.058396$, $r_L = 0.886532$, $r_T = 0.717928$.) Also, it can be deduced that on choosing 199 nodes in hidden layer (+ 1 bias node = 200 = number of data set) , RBF exactly learns the training data, but the testing data performance (i.e. generalization) is poor ($E_F = 0.000000$, $r_L = 1.0$, $r_T = 0.665$).

Training time of RBF network is less by several orders of magnitude than required for BP and its four variations, as tried in this work.

4.2.3 : Results of RBF on desulphurization problem

Networks of 8 different topologies (i.e. 8 different nodes in hidden layer) are shown in Table 4.15. The network 7-33-1 shows the best generalized results as compared to all other topologies ($r_L = 0.360351$, $r_T = 0.350241$). As in the case of dephosphorization problem, with increase in the number of nodes in the hidden layer, the performance during training improves but the generalization ability is adversely affected (for example for 99 centers (7-100-1 network) $E_F = 0.043975$, $r_L = 0.612558$, $r_T = 0.251655$.) Similar to the case of dephosphorization problem RBF has capability to exactly learn the training data (the network 7-349-1, $E_F = 0.000000$, $r_L = 1.0$, $r_T = 0.040$).

4.3 : Results of GA assisted BP (GA + BP) continued with BP

GA assisted BP is discussed in section 3.1 and 3.2.

4.3.1 : Dephosphorization problem

Table 4.16 shows the results of (GA + BP). Altogether 7 network topologies (on which BP and its variations are tested) are chosen for testing this algorithm. Population size for GA is chosen equal to 1000 and network is trained for 500 generations. The GA parameters are as follows

- (i) Multiple point crossover, with 4 crossover sites, crossover probability (p_c) = 0.9.
- (ii) Mutation probability (p_m) = 0.085.
- (iii) Precision for each weight is kept equal to 0.001 and the length of each weight (l_w) is calculated according to the Eq.(3.1) and then total chromosome length (L_T) is calculated from Eq.(3.2).
- (iii) Binary Tournament selection (n ary where $n = 2$) is taken for selection scheme.
- (iv) Search space for the weights is taken $[-20, 20]$ i.e. $w_{min} = -20$, $w_{max} = 20$ in Eq.(3.1).

After running GA for 500 generations, the best ever chromosome is decoded into real weights by Eq.(3.3). Then BP is started with the initial weights found by GA and run for 1000 epochs (see section 3.2). Learning rate and momentum factor are kept equal 0.8 and 0.5 respectively.

Table 4.16 shows the generations in which best ever chromosome are found. It can be deduced that in most cases the

learning stops in early stages (for example in generation numbers 67,47,57,193, best fitness has been achieved i.e. after this number of generations no learning take place at all).Learning error (after running GA plus 1000 number of epochs of BP), is higher than that of BP alone after 10,000 epochs. For example for 7-6-1 net after running BP with 10,000 number of epochs, $E_F = 0.076323$ (Table 4.3) while with GA+BP, $E_F = 0.093322$ (Table 4.16). This may be attributed to the two reasons:

- * The range of search space for weights are bound to [20,20].
- * The number of variable (weights) are large.

Since GA+BP learning performance is poor, therefore on testing for seen and unseen patterns also the network produces poor performance for learning data. For example for 7-6-1 network results of BP are $E_F = 0.0763$, $r_L = 0.832894$ whereas results of GA+BP on testing are $E_F = 0.0933$, $r_L = 0.747628$.

Training time for GA +BP is much higher than that of BP alone. In GA+BP network the objective function (network output) is evaluated (500 x 1000 =) 5,00,000 (in GA) + 1000 (in BP) = 5,01,000 number of times as compared to 10,000 epochs in BP alone. In some cases however BP+GA is known to give better results [3].

4.3.2 : Desulphurization problem

Table 4.17 summarizes the results of GA +BP on desulphurization problem. Parameters in learning with GA +BP are

kept the same as that for dephosphorization (section 4.3.1). The results, shown in Table 4.17, confirms the poor learning of GA +BP compared to that of BP alone (Table 4.3).

4.4 : Results of ANFIS

4.4.1 An example of learning of ANFIS network of type Fig (3.4 a)

The data set used to show one complete iteration of the learning of ANFIS network of type Fig (3.4 a) is given below; it comprises of 12 patterns taken from the data set of desulphurization data (Table 1.3).

| Initial S (%) | Tempera- ture (°C) | weight pig iron (tons) | weight CaD (kg) | Final S (%) |
|-----------------------|----------------------------|------------------------------|-----------------------|---------------------|
| 0.075 | 1417.0 | 230.0 | 1408.0 | 0.009 |
| 0.067 | 1369.0 | 196.0 | 1546.0 | 0.006 |
| 0.039 | 1383.0 | 215.0 | 1342.0 | 0.003 |
| 0.060 | 1413.0 | 220.0 | 1661.0 | 0.002 |
| 0.046 | 1435.0 | 240.0 | 1619.0 | 0.001 |
| 0.084 | 1429.0 | 230.0 | 1561.0 | 0.006 |
| 0.085 | 1371.0 | 244.0 | 1665.0 | 0.007 |
| 0.087 | 1382.0 | 215.0 | 1559.0 | 0.006 |
| 0.074 | 1374.0 | 228.0 | 1541.0 | 0.004 |
| 0.070 | 1382.0 | 246.0 | 1622.0 | 0.005 |
| 0.032 | 1400.0 | 180.0 | 1015.0 | 0.003 |
| 0.032 | 1433.0 | 190.0 | 1072.0 | 0.005 |

Each pattern consists of 4 inputs and one output. Let the number of MFs associated with each input be 2 (chosen by the

user ,i.e. $r_1 = r = 2$ (Eq.(3.7)). Thus, the number of square nodes in layer 1 will be $4 \times 2 = 8$. The number of nodes (i.e. number of rules fired) in layer 2 (h) will be equal to 2. The number of nodes in layer 3 will also be equal to 2. Since the number of nodes in output pattern is 1 therefore number of nodes in layer 4 will also be $(2 \times 1) = 2$. Output layer (layer 5) consists of single node. The learning algorithm proceeds as follows

- (i) Initial guess for premise parameters in layer 1 .
- (ii) Calculation of firing strength in layer 2 for each pattern .
- (iii) Calculation of normalized firing strength in layer 3 for each pattern .
- (iv) Calculation of consequent parameters.
- (v) Correction of premise parameters.

The details of each of the steps are given below.

Step (i) Initial guess for premise parameters in layer 1

Initial values for the premise parameters are calculated from Eqns.(3.16-3.18). The terms \max_1 and \min_1 are found out from the data set used for learning and are given below

| Input Number | Minimum | Maximum |
|-----------------|-----------|----------|
| 1 | 00.032000 | 0.087000 |
| 2 | 1369.0000 | 1435.000 |
| 3 | 180.00000 | 246.0000 |
| 4 | 1015.0000 | 1665.000 |

Initial premise parameters thus calculated from Eqns.(3.16-3.18) are shown below

| Input Number | I st MF associated with input number | | | 2 nd MF associated with input number | | |
|--------------|--------------------------------------|----------|--------|--------------------------------------|----------|--------|
| | a | b | c | a | b | c |
| 1 | 0.0275 | 2.123964 | 0.0320 | 0.0275 | 2.123964 | 0.0870 |
| 2 | 33.0000 | 2.123964 | 1369.0 | 33.000 | 2.123964 | 1435.0 |
| 3 | 33.0000 | 2.123964 | 180.00 | 33.000 | 2.123964 | 246.00 |
| 4 | 325.000 | 2.123964 | 1015.0 | 325.00 | 2.123964 | 1665.0 |

Step (ii) Calculation of Firing strength (w) for each pattern :

For the 1 st pattern the calculation of w are shown below .

Calculate the fuzzified membership value by the bell shape membership function (Eq.(3.6), a,b,c as shown above). These values are shown below

| Input number | Node 1 output | Node 2 output |
|---------------|---------------|---------------|
| (in Layer 1) | | |
| 1 | 0.130235 | 0.971327 |
| 2 | 0.169149 | 0.929223 |
| 3 | 0.146155 | 0.955856 |
| 4 | 0.308522 | 0.730507 |

Now the firing strength (output of layer 2) is calculated by Eq.(2.8) and is given below.

| Node in layer 2 | Firing Strength (w) |
|-----------------|---------------------|
| 1 | 0.000993 |
| 2 | 0.630235 |

Step (iii) Calculation of normalized firing strength (w) for each pattern for each pattern

Normalize firing strength (output of layer 3) is now calculated by Eq.(3.12) and is given below.

| Node Number in layer 3 | Normalize Firing Strength of node |
|---------------------------|--------------------------------------|
|---------------------------|--------------------------------------|

| | |
|---|----------|
| 1 | 0.001574 |
| 2 | 0.998426 |

Similarly for all patterns normalize firing strength is calculated and the results is given below.

| Pattern Number | Normalize Firing Strength (output of layer 3) | |
|-------------------|--|--------|
| | Node 1 | Node 2 |

| | | |
|----|----------|----------|
| 1 | 0.001574 | 0.998426 |
| 2 | 0.829736 | 0.170264 |
| 3 | 0.985387 | 0.014613 |
| 4 | 0.005288 | 0.994712 |
| 5 | 0.001497 | 0.998503 |
| 6 | 0.000070 | 0.999930 |
| 7 | 0.002893 | 0.997107 |
| 8 | 0.031697 | 0.968303 |
| 9 | 0.043733 | 0.956267 |
| 10 | 0.006208 | 0.993792 |
| 11 | 0.999903 | 0.000097 |
| 12 | 0.993900 | 0.006100 |

Step (iv) Calculation of Consequent parameters :

Number of nodes in layer 4 is 2 and each node is associated with 5 (4 input +1) consequent parameters. Thus, there are 10 unknown consequent parameters. There are 12 patterns to

train the network. Thus there will be 12 linear equations with 10 unknown consequent parameters. Pseudo inverse method (Eq.(3.19)) is used to calculate the consequent parameters (p's). The coefficient matrix A of consequent parameters is calculated by Eqns.(3.14,3.15). This coefficient matrix is given below :

| | | | | | |
|----|----------|-------------|------------|-------------|----------|
| A= | 0.000118 | 2.229872 | 0.361941 | 2.215709 | 0.001574 |
| | 0.074882 | 1414.770128 | 229.638059 | 1405.784291 | 0.998426 |
| | 0.055592 | 1135.908484 | 162.628242 | 1282.771743 | 0.829736 |
| | 0.011408 | 233.091516 | 33.371758 | 263.228257 | 0.170264 |
| | 0.038430 | 1362.790116 | 211.858189 | 1322.389252 | 0.985387 |
| | 0.000570 | 20.209884 | 3.141811 | 19.610748 | 0.014613 |
| | 0.000317 | 7.471250 | 1.163252 | 8.782552 | 0.005288 |
| | 0.059683 | 1405.528750 | 218.836748 | 1652.217448 | 0.994712 |
| | 0.000069 | 2.148328 | 0.359302 | 2.423793 | 0.001497 |
| | 0.045931 | 1432.851672 | 239.640698 | 1616.576207 | 0.998503 |
| | 0.000006 | 0.100314 | 0.016146 | 0.109580 | 0.000070 |
| | 0.083994 | 1428.899686 | 229.983854 | 1560.890420 | 0.999930 |
| | 0.000246 | 3.966219 | 0.705877 | 4.816743 | 0.002893 |
| | 0.084754 | 1367.033781 | 243.294123 | 1660.183257 | 0.997107 |
| | 0.002758 | 43.804639 | 6.814759 | 49.414929 | 0.031697 |
| | 0.084242 | 1338.195361 | 208.185241 | 1509.585071 | 0.968303 |
| | 0.003236 | 60.089502 | 9.971184 | 67.392957 | 0.043733 |
| | 0.070764 | 1313.910498 | 218.028816 | 1473.607043 | 0.956267 |
| | 0.000435 | 8.579165 | 1.527116 | 10.069035 | 0.006208 |
| | 0.069565 | 1373.420835 | 244.472884 | 1611.930965 | 0.993792 |
| | 0.031997 | 1399.864642 | 179.982597 | 1014.901865 | 0.999903 |
| | 0.000003 | 0.135358 | 0.017403 | 0.098135 | 0.000097 |
| | 0.031805 | 1424.258778 | 188.841010 | 1065.460858 | 0.993900 |
| | 0.000195 | 8.741222 | 1.158990 | 6.539142 | 0.006100 |

$$y = \begin{bmatrix} 0.009 \\ 0.006 \\ 0.003 \\ 0.002 \\ 0.001 \\ 0.006 \\ 0.007 \\ 0.006 \\ 0.004 \\ 0.005 \\ 0.003 \\ 0.005 \end{bmatrix} \quad \text{where } y \text{ is the target matrix.}$$

Consequent parameters found out by pseudo inverse of A is given below.

$$X = \begin{bmatrix} -3.197654e+00 \\ +1.564551e-04 \\ -1.934879e-03 \\ +2.834221e-04 \\ -5.310634e-02 \\ +1.122984e-01 \\ -4.649926e-05 \\ +1.074791e-05 \\ -2.122539e-05 \\ +9.397889e-02 \end{bmatrix}$$

The error E_F calculated by Eq.(3.20) is 0.00023877.

Step (v) Correction of premise parameters

Step size is chosen equal to 0.5. The sum of square of all derivatives (denominator term in Eq.(2.24)) comes out to be $4.330e-9$. Thus learning rate (η) calculated is $7.597e+03$. Premise parameters are corrected by Eqns.(3.21-3.23). The corrected premise parameters are shown below.

| Input Number | 1 st MF associated with input number | | | 2 nd MF associated with input number | | |
|-----------------|---|---|---|---|---|---|
| | a | b | c | a | b | c |

| | | | | | | |
|---|----------|--------|-----------|----------|--------|-----------|
| 1 | -0.261 | +2.135 | +0.294 | -0.1110 | +2.122 | +0.365 |
| 2 | +32.999 | +2.123 | +1368.999 | +32.999 | +2.129 | +1434.999 |
| 3 | +33.000 | +2.113 | +180.000 | +33.000 | +2.123 | +245.999 |
| 4 | +324.999 | +2.125 | +1015.000 | +324.999 | +2.123 | +1665.000 |

Thus, the premise parameters and consequent parameters are adjusted one after the other till error minimize to a specified acceptance level.

4.4.2 : Dephosphorization problem

(a) Results of ANFIS network of type Fig (3.4 a)

Table 4.18 shows the results of ANFIS (network 3. (a)). Network is trained for 8 topologies by varying the number of membership function (MF) associated with each input. Various parameters are as follows

- (i) Number of membership function associated with each input ($r_i = r$ (Eq.(3.7)) varies with different topologies.
- (ii) number of epochs = 1000.
- (iii) Initial premise parameters are generated as per the Eqns.(3.16-3.18).
- (iv) Step size = 0.5

Consequent parameters are evaluated by solving equation $Ax=B$ while using pseudo inverse method. No data preprocessing is required in this case, but error shown in Tables (4.18-4.20) are on the scaled basis (for the sake of comparison with other algorithms). It can be deduced from Table 4.18 that with increasing number of MFs associated with each input, the error level decreases to very low levels (i.e. problem of over learning occurs). Therefore, prediction performance on unseen data set is poor (due to over learning). For example, with 2 MFs associated with each input $E_F = 0.071093$, $r_L = 0.826277$ and $r_T = 0.752207$, whereas with 10 MFs associated with each input $E_F = 0.036695$, $r_L = 0.941677$ and $r_T = 0.223731$.

(b) Results of ANFIS network of type Fig (3.4 b)

If 2 membership functions are associated with each input then the number of nodes in layer 2 will be $2^7 (=128)$. Thus the number of nodes in layer 4 will also be 128. The number of consequent parameters will then be $128 \times 8 = 1024$. Thus the minimum number of data set required will be 1024. In the dephosphorization problem, only 200 data sets were available therefore ANFIS network of type Fig (3.4 b) could not be implemented in this problem.

4.4.3 : Desulphurization problem

(a) Results of ANFIS network of type Fig (3.4 a)

The results for 8 different topologies (i.e. different

MFs associated with each input) are shown in Table 4.18. These results confirm the overlearning of bigger networks.

(b) Results of ANFIS network of type Fig (3.4 b)

In desulphurization problem, this type of network is run for 2 MFs associated with each input. The results are shown in Table 4.19. The results obtained are better than that of BP. For example in BP (Table 4.9) $E_F = 0.0531$ $r_L = 0.310$, $r_T = 0.442$ while in ANFIS of type Fig 3.4 (b) $E_F = 0.0486$ $r_L = 0.486$, $r_T = 0.378$. Higher topologies could not be tried because of less data available.

Training time required for ANFIS is less than that of BP while results are better or comparable to BP.

4.5 Results of GA assisted ANFIS

GA assisted ANFIS is discussed in chapter 3 (section 3.4) and is implemented for both type of networks (Fig (3.4 a,b)). The procedure for the calculation of search space for premise parameters (a,b,c) is given in section 3.4 . The unknowns \min_1 and \max_1 are given in Table 4.1,4.2 (for the two problems of dephosphorization and desulphurization). Other parameters are as follows :

- (i) Binary tournament selection scheme is used .
- (ii) Multiple point crossover with probability (p_c)= 0.9.

- (iii) Mutation probability (p_m) = 0.05
- (iv) Number of Generation = 100
- (v) population size = 50

4.5.1 : Dephosphorization problem

Only the network of type Fig (3.4 a) is tested for dephosphorization problem because of less data available. The same 8 different network topologies, which were tested by ANFIS, are selected for training by GA + ANFIS. The results are shown in Table (4.21). GA +ANFIS is more successful than ANFIS in small networks but in higher networks ANFIS is slightly better than GA +ANFIS . The success of GA + ANFIS may be attributed to the fact that the range of premise parameters are selected in the proper range. One example of comparison of ANFIS and GA +ANFIS in network 7-14-2-2-2-1 is given below

| | | | | | | |
|------------|---|----------------|---|----------------|---|----------------|
| ANFIS | : | $E_F = 0.0710$ | , | $r_L = 0.8262$ | , | $r_T = 0.7522$ |
| GA + ANFIS | : | $E_F = 0.0655$ | , | $r_L = 0.8549$ | , | $r_T = 0.8609$ |

In case of bigger networks overlearning can be seen in the results shown in Table 4.21. Learning time for GA +ANFIS is more than that of ANFIS.

4.5.2 : Desulphurization problem

(a) Results of GA + ANFIS learning of type Fig (3.4 a)

The results (listed in Table 4.22) conform with the

results obtained in the case of dephosphorization problem. i.e. GA + ANFIS gives better results than ANFIS for smaller network. One example of comparison of ANFIS and GA + ANFIS in network 4-16-4-4-4-1 is given below.

ANFIS : $E_F = 0.0469$, $r_L = 0.5357$, $r_T = 0.3040$
 GA + ANFIS : $E_F = 0.0342$, $r_L = 0.7882$, $r_T = 0.3424$

(b) Results of GA + ANFIS learning of type Fig (3.4 b)

The only network tried with the algorithm is 4-8-16-16-16-1 (Table 4.23). The results are compared with the same network topology trained by ANFIS (Table 4.23). The results of GA + ANFIS are better than that of ANFIS (Table 4.20). For the 4-8-16-16-16-1 network :

ANFIS : $E_F = 0.0486$, $r_L = 0.4865$, $r_T = 0.3780$
 GA + ANFIS : $E_F = 0.0242$, $r_L = 0.8997$, $r_T = 0.2304$

4.6 Comparison of the results of various algorithms

Best results for all algorithms for dephosphorization problem are summarized in Table (4.24). Amongst the backpropagation and its 4 variations, the variation 1 gives the best results. The learning error vs iteration plot for the dephosphorization problem for 7-6-1 network is shown in Fig (4.1); it can be seen from the Fig (4.1) that convergence of BP is

fastest (amongst all the 4 variations tried).

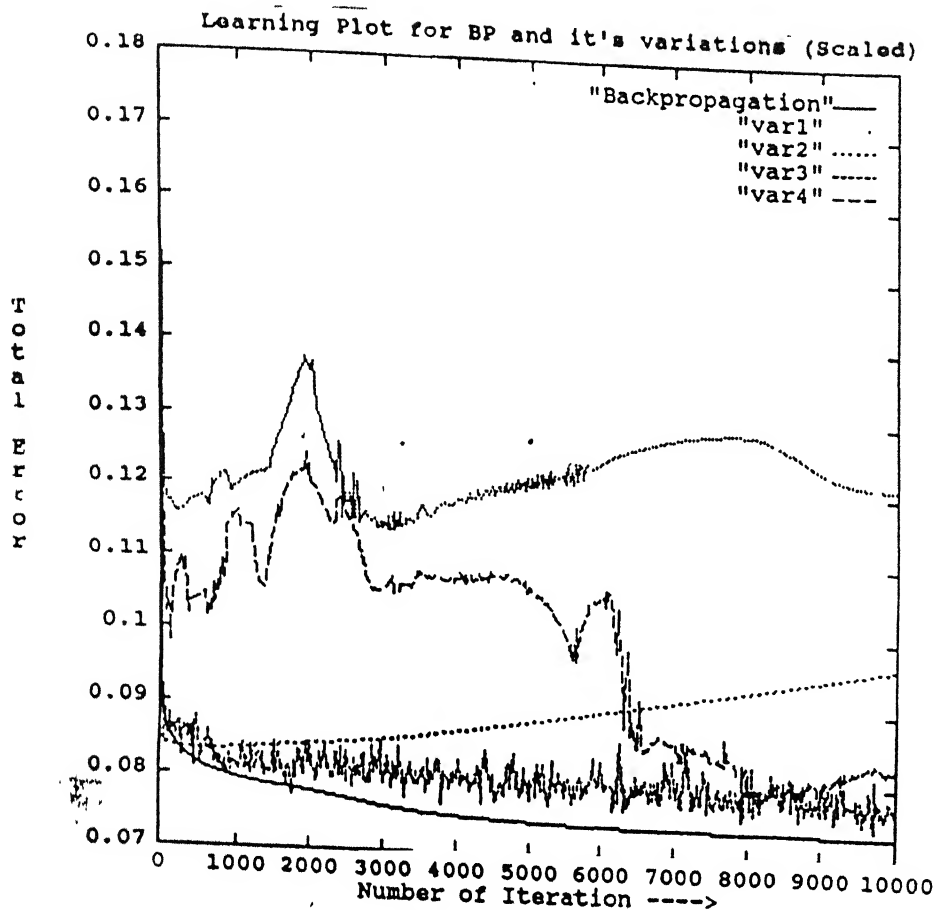


Fig 4.1 : Learning Plot for different variations of BP

Radial Basis function gives more generalized results than BP and other algorithms tried. GA assisted BP does not work well in both of the problems. When using ANFIS, the problems of overlearning were encountered in most of the cases because the number of adjustable parameters are vary large. GA assisted ANFIS gives better results than ANFIS for smaller size networks. It can be concluded that GA + ANFIS gives the best results among all the algorithms tried out for example in dephosphorization problem for 7-14-2-2-2-1 network $E_F = 0.0655$, $r_L = 0.8549$, $r_T = 0.8609$, standard deviation for learning data = 1.549, and 1.244 for

testing data , root mean percentage error for learning data= 13.37
while for testing data root mean percentage error = 9.144.

Learning of various algorithms is shown in Fig 4.2. Best learning is achieved in case GA + ANFIS. Moreover, the results are also better than other methods as shown above.

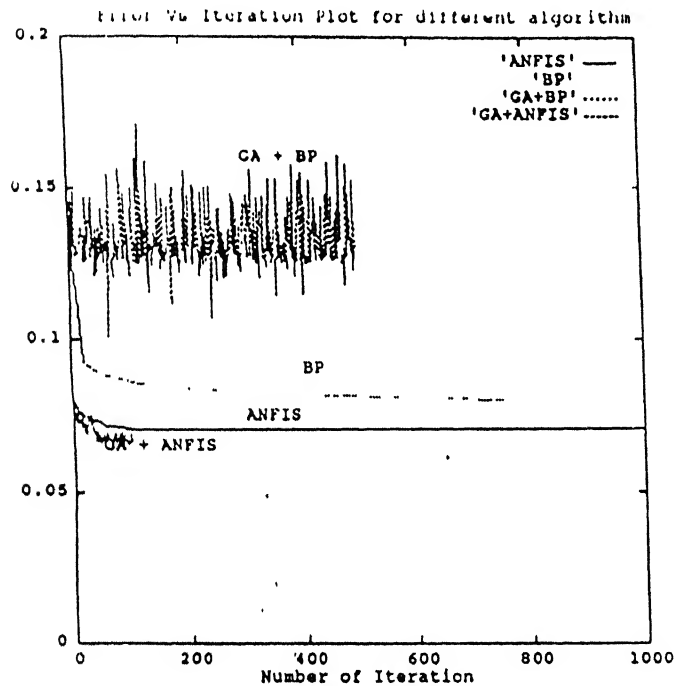


Fig 4.2 : Learning plot for different algorithms

Training time for RBF is the lowest and that in case of GA + NN was highest. A rough estimation of the training time required for various algorithms is given below. Training time increases in order RBF -> ANFIS -> BP and its variations -> GA +ANFIS -> GA + NN.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

5.1 Conclusions :

Modeling with modern artificial methods like neural networks, genetic adaptive search and GA assisted fuzzy neural networks have now become a good tool to model complex processes. The advantage of the methods is that they can be applied without any prior knowledge of the theory of processes. Results obtained in this work reveals the following:

1. Backpropagation :

Backpropagation and its four variations are good tools for the modeling of dephosphorization and desulphurization in iron and steel making. Random presentation of patterns in backpropagation (variation 1) gives better generalization results than other variations of backpropagation. Convergence of simple

backpropagation method, is ,however, fastest compared to the 4 variations tried.

2. ANFIS Vs Backpropagation :

The training time required of ANFIS is less than that of backpropagation. The results of ANFIS obtained on small sized networks (e.g. for 2 membership functions associated with each input) is comparable to backpropagation but for larger networks (e.g. 6 membership functions associated with each input), the problem of overlearning occurs.

3. Radial basis function

Radial basis function offer the fastest method of learning among all the algorithms tried in this work. Its performance is also comparable to BP provided the data set is large.

4. Integration of GA with neural networks and fuzzy logic :

Integration of genetic algorithms with neural networks increases the training time. In some problems it may not yield satisfactory results. For example (GA + BP) gives poorer results than BP. Integration of GA with ANFIS, however, gives good generalized results compared to the other algorithms tried in this work.

5. Overlearning

The problems of overlearning increases with the number of adjustable parameters in all algorithms.

6. Best results :

The results obtained for dephosphorization problem with GA + ANFIS are good (correlation coefficient for learning data = 0.85 while for testing data = 0.86) but the results obtained for desulphurization problem are not satisfactory. These results can ,however, be improved by incorporating the theoretical models for preprocessing of data.

5.2 Recommendations for future work :

The following aspects need to be studied in detail for critical comparison of various algorithms :

1. Effect of nonlinearity in output layer of radial basis function needs to be studied.
2. Other variations of backpropagation like BP based on second derivative methods and Guass Newton method need to be studied.
3. Effect of data preprocessing of data and noise injection in

inputs in various algorithms should be studied.

4. Meaningful linguistic rules derived from theory of metallurgical processes can be incorporated in ANFIS and GA +ANFIS and obtain better results (compared to that in the present work).

Modern methods and algorithms developed in the present work can be extended to other metallurgical processes e.g ladle furnace operation, continuous casting and rolling mills.

R E F E R E N C E S

1. Peter J W. Melsa, " Neural Networks : A Conceptual Overview," TRC-89-08, August 1989.
2. Lippmann. R. P., An Introduction to Computing with Neural nets, IEEE ASSP Magazine., V-4, pp. 4-22, 1987.
3. Janson D.J. and Frenzel F., (1993), Training Product Unit Neural Networks with Genetic Algorithms , IEEE EXPERT, pp. 26-32, October 1993.
4. Zadeh, L. A., (1988), Fuzzy Logic , Computers, Vol. 21, pp. 83-92.
5. Jang Roger S.J., (1993), ANFIS: Adaptive - Network - Based Fuzzy Inference System , IEEE Trans on SMC, Vol. 23, No. 3, pp. 655-685.
6. Fu LiMin, (1994), " Neural Networks in Computer Intelligence ", McGraw Hill Book Company, Singapore.
7. Y. H. Pao (1989), " Adaptive Pattern Recognition and Neural Networks," Addison-wesley Publishing Company, Inc., Reading, Massachusetts
8. Reuter M. A., " Modeling of Metal-Slag Equilibrium Process Using Neural Nets," Metallurgical Transaction B , Vol 23B, pp 643 - 650, Oct 1992-93.
9. Phiroz bhagat, " An Introduction to Neural Nets," Chemical Engineering Process, pp. 55-60, August 1990.
10. Wassserman P.D. (1989), " Neural Computing : Theory and Practice," Van Nostrand Reinhold, New York.
11. Christian Cachin, " Pedagogical Pattern Selection Strategies, " Neural networks, vol 7 No 1, pp 175 - 181, 1994.
12. Ooyen A. Van, Improving the Convergence of the Back-Propagation Algorithm, Neural Networks, Vol 5, pp. 465-471, 1992.
13. Abhay Bulsari et.al , "Investigation of The Systemmetric Logarithmoid As an Activation Function For Neurans In Feed Forward Neural Networks," Artificial Neural Networks, Elsevier Science Publisher, B.V. (North Holland), pp. 653-657, 1991.
14. Hush D. and Horne B. G., Progress in Supervised neural Networks (What New Since Lippmann), IEEE Signal Processing Magazine, 1993.

15. Leonard J.A. and Kramer M. A., Radial Basis Function Networks for Classifying Process Faults , IEEE Control Systems, April 1991.
16. Uehara K et.al., (1993)," Multistage Fuzzy Inference Formulation as Linguistic Truth Value Propagation and Its Learning Algorithm on Back-Propagating Error Information," IEEE Transation On Fuzzy Systems, Vol 1, No. 3, August 1993.
17. Adhikari Amitabh Prasad, (1995), M. Tech Thesis, Dept of NET, I.I.T. Kanpur.
18. Davis L., (1991), Handbook of Genetic Algorithms}, Van Nostrand Reinhold, New York.
19. Goldberg D.E., (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison--Wesley, Reading, Mass.

APPENDIX A

AN INTRODUCTION TO GENETIC ALGORITHMS

Genetic Algorithms (GAs) is a search procedure developed by John Holland and his associates during early 1970. It is based on the natural adaptation and evolution [18]. Some of the general features of the theory of evolution that are widely accepted :

- Evolution is a process that operates on chromosomes rather than on living beings they encode.
- Natural selection is a the link between chromosomes and the performance of their decoded structures. Processes of natural selection cause those chromosomes that encode successful structures to produce more often than those that do not.
- The process of reproduction is the point at which evolution takes place. Mutations may cause the chromosomes of biological children to be different from those of their biological parents, and recombination process may create quite different chromosomes in their children by combining material from chromosomes of two parents.

Genetic algorithms is based on integration of the above techniques. Genetic algorithms is different from other classical optimization techniques by following characteristics.

1. Genetic algorithms works with coding of parameter set (not the parameters themselves).
2. Genetic algorithms search the optimum from a population of points (not from a single points) .
3. Genetic algorithms used only object function information.
4. Genetic algorithms used probabilistic translation rules (not deterministic rules.)

A genetic algorithms operates through a simple cycle of states.

1. Creation of a "population" of strings.
2. Evaluation of each string.
3. Selection of the "best" strings.
4. Genetic manipulation to create the new population of strings.

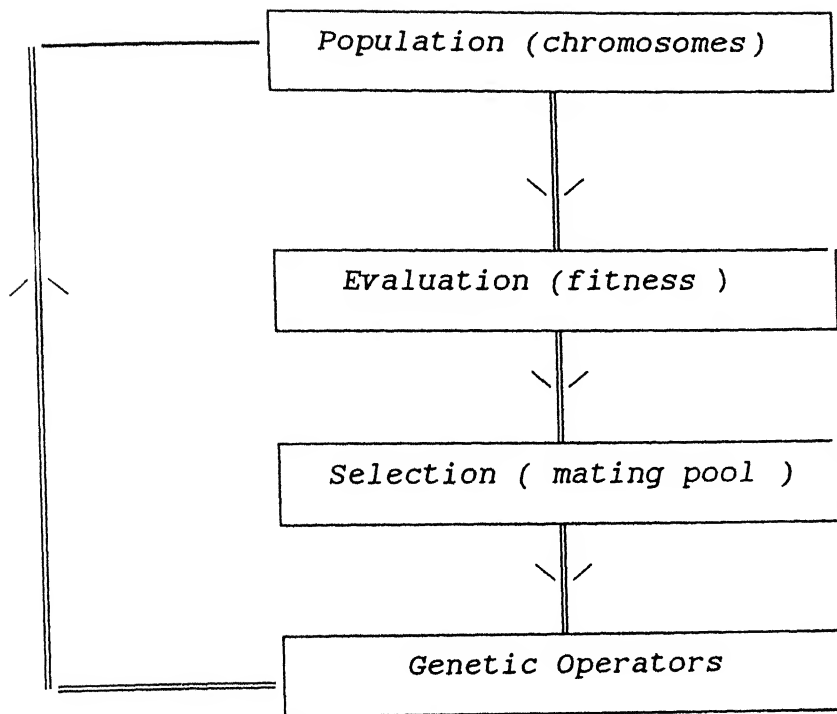
Each cycle of genetic algorithms (Fig 1) produces a new generation of possible solution for a given problem. In the first stage, an initial random population is created as a starting point for this search. Each element of population is treated as a starting point for the search. Each element of population is encoded into a string (the chromosomes) which in turn are

manipulated by genetic operators. In the next stage, the performance (or the fitness) of each individual is evaluated with respect to the constraints imposed by the problems. Based on each individual's fitness, a selection mechanism chooses "mates" for genetic manipulation process. The selection policy is ultimately responsible for assuring survival of the best fitted individuals. The combined evolution and selection process is called reproduction [19].

The manipulation process uses genetic operators to produce a new population of individuals (offspring). It comprises of two operators : crossover and mutation. Crossover recombines a population's genetic material. The selection process associated with recombination assures that special genetic structures, called building blocks, are retained for future generations. The building blocks then represent the most fitted genetic structures in a population.

1. Selection or Reproduction :

Selection or reproduction is a process in which individual strings are copied according to the fitness of objective function. Copying means the string having higher fitness have a higher probability of contribution of one or more offsprings in the next generation. Reproduction may be done in many ways as *Roulette wheel* ,*Tournament Selection* ,*Stochastic remainder*. Roulette wheel and tournament selection is described below.



GAs Operators : Crossover and mutation

FigA1 : Cycle of genetic algorithms

(i) Roulette Wheel Parent Selection :

1. Sum the fitnesses of all the population members : call the result total fitness.
2. Generate 'n', a random number between 0 and total fitness.
3. Return the first population whose fitness, added to the fitnesses of the proceeding population members is greater than or equal to n.

(ii) Tournament Selection :

1. Select the half of the population (call p randomly).
2. Select any number m between 2 to p.
3. Select m samples out of p samples, evaluate them and select fittest among them .Do it for m times.
4. Do the same process once again.

2. Crossover :

The function of this operator is to alter chromosomes created during reproduction such that they differ from their parents. In nature, crossover occurs when two parents exchange parts of their chromosomes. p_c called probability of crossover is given to this operator which decides whether the crossover is done or not on a particular pair of chromosome. There are different types of crossover operators — single point crossover, multiple point crossover etc. Single point crossover in genetic algorithms can be shown as follows :

| | | | | | | | | | | | |
|-----------|---|---|--|---|---|-----|-----------|---|---|---|---|
| Parent 1: | 1 | 1 | | 1 | 1 | - > | child 1 : | 1 | 1 | 0 | 0 |
| Parent 2: | 0 | 0 | | 0 | 0 | | child 2 : | 0 | 0 | 1 | 1 |

In single point crossover swap is done at single site, selected randomly. In the same way in multiple point crossover swap is done at multiple sites (chosen randomly) The probability of crossover p_c is generally selected close to 1 (0.6- 0.9). A lower probability of crossover means that lesser number of parents

will crossover and then the children will be same as the parents. The steps involved in single point crossover is given below.

(i) Single Point Crossover :

1. Select the two chromosomes.
2. Generate a number 'n', randomly between 0 to length of the chromosomes.
3. Swap the chromosomes at this point.

(ii) Multiple Point Crossover :

1. Select a number 'm' that is the number of sites.
2. Do the single point crossover 'm' times.

3. Mutation :

The mutation of a bit involves changing a 0 to 1 or vice versa. Just as P_c controls the probability of crossover, P_m (the mutation rate), controls the probability of flipping. The bits of a string are independently mutated — that is, mutation of a bit does not affect the probability of mutation of other bits. Mutation is a secondary operator whose role is to restore lost genetic materials. For example, suppose all the strings in a population have converged to a 0 at a given position and optimal solution has 1 at that position, then crossover cannot regenerate a 1 at that position, while a mutation could. Mutation is done after crossover.

TABLES

Table 1.1 : Learning data used for dephosphorization problem
(200 examples)

| Hot Metal Weight (tons) | T (°C) | C % x 100 | Mn % x 100 | Init P % x 100 | Ore (kg) | O ₂ blown (Nm ³) | Final P % x 100 |
|----------------------------------|-----------|-----------------|------------------|-------------------------|-------------|---|--------------------------|
| 315.0 | 1613 | 447 | 314 | 26 | 1000 | 14100 | 11 |
| 316.1 | 1606 | 384 | 330 | 23 | 900 | 13860 | 11 |
| 316.1 | 1620 | 223 | 312 | 22 | 500 | 13220 | 14 |
| 315.5 | 1606 | 543 | 349 | 27 | 1044 | 14240 | 14 |
| 311.0 | 1611 | 496 | 333 | 25 | 500 | 15020 | 12 |
| 314.3 | 1636 | 257 | 314 | 30 | 1000 | 14040 | 16 |
| 314.9 | 1642 | 425 | 338 | 32 | 1064 | 14500 | 16 |
| 316.2 | 1642 | 332 | 334 | 25 | 1020 | 13940 | 14 |
| 314.3 | 1616 | 371 | 306 | 38 | 1020 | 14150 | 16 |
| 313.5 | 1602 | 529 | 303 | 22 | 880 | 14850 | 9 |
| 312.7 | 1617 | 438 | 330 | 26 | 824 | 13850 | 13 |
| 313.2 | 1613 | 413 | 298 | 24 | 514 | 13780 | 13 |
| 314.8 | 1620 | 569 | 314 | 30 | 510 | 14360 | 17 |
| 315.2 | 1620 | 524 | 341 | 29 | 1014 | 14010 | 18 |
| 314.7 | 1588 | 465 | 283 | 17 | 1044 | 13850 | 9 |
| 319.0 | 1633 | 278 | 290 | 22 | 1064 | 12330 | 13 |
| 314.6 | 1638 | 259 | 334 | 28 | 1038 | 14530 | 13 |
| 315.9 | 1619 | 368 | 315 | 25 | 1038 | 13940 | 14 |
| 315.3 | 1616 | 382 | 333 | 27 | 1030 | 13760 | 12 |
| 314.0 | 1598 | 198 | 260 | 17 | 494 | 14370 | 8 |
| 315.9 | 1597 | 448 | 321 | 26 | 604 | 14090 | 13 |
| 312.2 | 1616 | 389 | 326 | 29 | 1010 | 13670 | 13 |
| 314.2 | 1604 | 444 | 321 | 26 | 1210 | 13860 | 26 |
| 314.4 | 1685 | 388 | 167 | 11 | 1000 | 14370 | 11 |
| 314.1 | 1602 | 517 | 343 | 26 | 1000 | 14100 | 11 |
| 314.7 | 1672 | 364 | 142 | 8 | 800 | 14030 | 8 |
| 315.9 | 1661 | 325 | 140 | 7 | 1000 | 14180 | 7 |
| 313.6 | 1597 | 540 | 328 | 19 | 850 | 14170 | 10 |
| 315.1 | 1599 | 423 | 306 | 24 | 758 | 14060 | 10 |
| 313.0 | 1615 | 425 | 385 | 30 | 400 | 14060 | 19 |
| 315.3 | 1638 | 214 | 334 | 27 | 1004 | 13500 | 17 |
| 314.0 | 1615 | 473 | 292 | 22 | 918 | 13820 | 11 |
| 321.0 | 1647 | 225 | 357 | 33 | 1050 | 13570 | 18 |
| 320.5 | 1622 | 406 | 339 | 29 | 1034 | 15010 | 12 |
| 314.4 | 1611 | 428 | 321 | 30 | 1024 | 13920 | 14 |
| 313.4 | 1610 | 308 | 330 | 32 | 814 | 14430 | 9 |
| 312.7 | 1578 | 499 | 265 | 19 | 1000 | 13930 | 8 |
| 315.0 | 1589 | 438 | 309 | 22 | 1010 | 13930 | 11 |
| 314.4 | 1616 | 405 | 322 | 29 | 1034 | 14000 | 15 |
| 314.7 | 1632 | 523 | 358 | 34 | 1010 | 14400 | 15 |
| 315.5 | 1601 | 572 | 287 | 24 | 1000 | 14510 | 10 |
| 324.6 | 1619 | 445 | 300 | 23 | 1024 | 14440 | 11 |

Table 1.1 continues

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 314.6 | 1580 | 573 | 304 | 19 | 1004 | 14240 | 8 |
| 314.9 | 1607 | 490 | 302 | 27 | 1020 | 14210 | 13 |
| 315.2 | 1601 | 542 | 331 | 23 | 504 | 13660 | 14 |
| 319.4 | 1616 | 623 | 332 | 26 | 1054 | 14510 | 14 |
| 321.1 | 1660 | 454 | 311 | 23 | 1001 | 14419 | 9 |
| 316.9 | 1584 | 415 | 303 | 20 | 510 | 14310 | 8 |
| 320.5 | 1629 | 250 | 314 | 23 | 1010 | 13880 | 12 |
| 314.8 | 1615 | 340 | 320 | 24 | 504 | 13730 | 14 |
| 316.0 | 1600 | 343 | 295 | 17 | 786 | 13590 | 8 |
| 313.7 | 1582 | 472 | 280 | 17 | 544 | 14090 | 8 |
| 314.1 | 1618 | 305 | 302 | 19 | 1024 | 13480 | 10 |
| 315.3 | 1609 | 400 | 331 | 19 | 1000 | 13830 | 10 |
| 313.8 | 1619 | 212 | 280 | 17 | 498 | 13790 | 9 |
| 314.8 | 1607 | 535 | 323 | 23 | 1270 | 13690 | 11 |
| 313.8 | 1606 | 204 | 286 | 14 | 1010 | 13330 | 7 |
| 316.1 | 1581 | 356 | 279 | 14 | 918 | 13940 | 7 |
| 314.5 | 1605 | 419 | 335 | 24 | 510 | 14200 | 13 |
| 314.4 | 1623 | 217 | 264 | 18 | 550 | 13240 | 9 |
| 315.9 | 1622 | 443 | 326 | 24 | 1504 | 14030 | 10 |
| 315.5 | 1615 | 416 | 333 | 22 | 874 | 13530 | 12 |
| 317.2 | 1640 | 290 | 325 | 21 | 1200 | 13690 | 11 |
| 316.3 | 1597 | 366 | 300 | 18 | 918 | 13730 | 9 |
| 316.0 | 1591 | 480 | 316 | 23 | 1000 | 13970 | 10 |
| 317.0 | 1601 | 328 | 304 | 19 | 1000 | 13490 | 9 |
| 316.8 | 1605 | 308 | 304 | 22 | 1014 | 13440 | 9 |
| 316.4 | 1600 | 469 | 317 | 21 | 1010 | 13810 | 10 |
| 322.0 | 1608 | 490 | 334 | 35 | 1000 | 13950 | 13 |
| 315.6 | 1605 | 306 | 311 | 21 | 500 | 13740 | 11 |
| 315.3 | 1612 | 336 | 314 | 22 | 900 | 13700 | 11 |
| 313.4 | 1628 | 239 | 299 | 18 | 840 | 14130 | 9 |
| 316.7 | 1624 | 390 | 336 | 26 | 1004 | 14180 | 12 |
| 316.0 | 1605 | 412 | 335 | 22 | 1040 | 14160 | 11 |
| 316.4 | 1628 | 457 | 339 | 35 | 1008 | 14900 | 10 |
| 316.5 | 1621 | 222 | 316 | 24 | 1014 | 13860 | 13 |
| 315.4 | 1604 | 87 | 198 | 14 | 800 | 14620 | 13 |
| 317.1 | 1641 | 508 | 375 | 42 | 1004 | 14160 | 21 |
| 317.0 | 1611 | 622 | 353 | 38 | 478 | 13940 | 21 |
| 313.2 | 1603 | 513 | 350 | 28 | 1000 | 14380 | 14 |
| 320.4 | 1608 | 497 | 359 | 28 | 1004 | 14810 | 14 |
| 315.4 | 1639 | 477 | 357 | 34 | 1000 | 14560 | 16 |
| 315.2 | 1627 | 276 | 332 | 27 | 1010 | 13930 | 12 |
| 313.8 | 1630 | 415 | 355 | 26 | 1004 | 14240 | 11 |
| 314.1 | 1619 | 269 | 316 | 20 | 500 | 13430 | 15 |
| 314.4 | 1613 | 498 | 336 | 29 | 518 | 14130 | 13 |
| 315.6 | 1617 | 419 | 357 | 27 | 504 | 14240 | 14 |
| 314.4 | 1624 | 527 | 340 | 29 | 994 | 14810 | 11 |
| 315.6 | 1617 | 428 | 334 | 29 | 978 | 14330 | 13 |
| 316.6 | 1618 | 290 | 321 | 23 | 880 | 13700 | 13 |
| 315.9 | 1634 | 407 | 362 | 29 | 1014 | 14160 | 15 |
| 318.3 | 1625 | 445 | 358 | 37 | 1014 | 14170 | 17 |
| 317.1 | 1627 | 305 | 330 | 30 | 918 | 13710 | 15 |
| 316.4 | 1617 | 273 | 328 | 23 | 520 | 13820 | 15 |
| 315.3 | 1632 | 357 | 323 | 27 | 1024 | 13540 | 14 |
| 315.6 | 1616 | 503 | 316 | 25 | 1014 | 13840 | 11 |

Table 1.1 continues

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 312.4 | 1624 | 283 | 328 | 26 | 1010 | 13610 | 12 |
| 315.3 | 1601 | 481 | 305 | 21 | 1010 | 14000 | 11 |
| 316.1 | 1594 | 536 | 328 | 24 | 520 | 14170 | 12 |
| 315.6 | 1644 | 363 | 386 | 30 | 1004 | 14090 | 18 |
| 313.0 | 1635 | 261 | 335 | 26 | 524 | 14480 | 12 |
| 316.0 | 1616 | 509 | 329 | 25 | 1014 | 13720 | 12 |
| 313.5 | 1606 | 713 | 338 | 33 | 918 | 14470 | 13 |
| 315.2 | 1587 | 601 | 303 | 19 | 478 | 14230 | 12 |
| 315.2 | 1593 | 401 | 282 | 17 | 1008 | 13540 | 9 |
| 312.9 | 1589 | 639 | 325 | 22 | 530 | 14290 | 9 |
| 315.6 | 1614 | 432 | 314 | 25 | 1024 | 14030 | 11 |
| 314.9 | 1601 | 534 | 325 | 26 | 918 | 14360 | 11 |
| 316.3 | 1612 | 228 | 302 | 21 | 1000 | 13320 | 12 |
| 323.9 | 1608 | 535 | 337 | 22 | 1014 | 14360 | 12 |
| 314.1 | 1621 | 516 | 355 | 31 | 1000 | 14240 | 15 |
| 316.6 | 1610 | 330 | 317 | 23 | 500 | 13330 | 15 |
| 315.9 | 1635 | 412 | 341 | 35 | 918 | 14090 | 15 |
| 316.2 | 1599 | 582 | 329 | 28 | 1010 | 14020 | 16 |
| 313.7 | 1570 | 564 | 317 | 28 | 400 | 14540 | 10 |
| 316.9 | 1597 | 423 | 314 | 21 | 576 | 13580 | 14 |
| 315.3 | 1598 | 490 | 317 | 21 | 1020 | 13980 | 10 |
| 316.2 | 1609 | 409 | 315 | 21 | 1000 | 13530 | 12 |
| 315.4 | 1613 | 286 | 312 | 18 | 1020 | 13630 | 10 |
| 316.2 | 1619 | 317 | 338 | 30 | 810 | 13830 | 14 |
| 314.7 | 1598 | 418 | 308 | 23 | 584 | 13910 | 12 |
| 314.0 | 1598 | 361 | 319 | 29 | 514 | 13880 | 10 |
| 317.4 | 1616 | 315 | 306 | 24 | 1014 | 13730 | 13 |
| 314.8 | 1610 | 447 | 322 | 27 | 1020 | 14440 | 13 |
| 315.8 | 1628 | 379 | 372 | 30 | 1014 | 14180 | 16 |
| 315.0 | 1635 | 405 | 337 | 28 | 928 | 14530 | 13 |
| 315.0 | 1619 | 403 | 313 | 25 | 504 | 14210 | 14 |
| 316.0 | 1637 | 409 | 352 | 38 | 998 | 13530 | 17 |
| 314.7 | 1629 | 336 | 320 | 26 | 984 | 13420 | 13 |
| 312.5 | 1639 | 181 | 256 | 18 | 514 | 14190 | 12 |
| 315.1 | 1621 | 445 | 306 | 26 | 964 | 14190 | 11 |
| 315.6 | 1594 | 436 | 301 | 19 | 1512 | 14100 | 12 |
| 315.0 | 1591 | 367 | 301 | 18 | 514 | 13440 | 10 |
| 316.9 | 1621 | 339 | 345 | 23 | 1004 | 13690 | 11 |
| 316.7 | 1631 | 408 | 342 | 38 | 1004 | 13830 | 14 |
| 316.5 | 1599 | 448 | 315 | 25 | 1000 | 13840 | 11 |
| 314.2 | 1608 | 322 | 328 | 20 | 300 | 13690 | 12 |
| 315.1 | 1642 | 145 | 313 | 24 | 414 | 13850 | 16 |
| 315.0 | 1606 | 305 | 338 | 25 | 514 | 13890 | 14 |
| 314.0 | 1623 | 230 | 304 | 19 | 1004 | 13280 | 11 |
| 315.9 | 1609 | 514 | 351 | 25 | 1010 | 14030 | 12 |
| 314.6 | 1606 | 367 | 345 | 24 | 1014 | 13510 | 13 |
| 317.2 | 1611 | 292 | 304 | 27 | 814 | 14120 | 14 |
| 316.0 | 1619 | 315 | 335 | 27 | 808 | 13900 | 13 |
| 315.0 | 1599 | 556 | 340 | 24 | 1000 | 13640 | 14 |
| 000.1 | 1607 | 312 | 307 | 32 | 510 | 13630 | 15 |
| 317.3 | 1611 | 327 | 332 | 28 | 1020 | 13740 | 13 |
| 316.4 | 1606 | 577 | 321 | 25 | 1008 | 14520 | 13 |
| 315.6 | 1609 | 590 | 351 | 25 | 1008 | 14220 | 14 |
| 316.0 | 1609 | 457 | 332 | 24 | 1024 | 14000 | 13 |

Table 1.1 continues

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 309.8 | 1591 | 499 | 298 | 23 | 1008 | 13830 | 12 |
| 315.9 | 1612 | 462 | 321 | 23 | 998 | 14110 | 12 |
| 315.6 | 1615 | 255 | 323 | 20 | 1412 | 14130 | 9 |
| 315.0 | 1664 | 419 | 314 | 20 | 500 | 14070 | 10 |
| 314.5 | 1605 | 442 | 314 | 19 | 510 | 13770 | 12 |
| 314.2 | 1636 | 303 | 321 | 24 | 1334 | 14210 | 11 |
| 315.8 | 1589 | 443 | 286 | 20 | 1010 | 13760 | 9 |
| 313.2 | 1611 | 640 | 358 | 29 | 998 | 13900 | 15 |
| 315.7 | 1611 | 471 | 335 | 25 | 1004 | 13600 | 15 |
| 315.6 | 1611 | 337 | 320 | 25 | 998 | 13770 | 13 |
| 316.5 | 1598 | 444 | 323 | 26 | 994 | 13690 | 13 |
| 316.8 | 1601 | 347 | 304 | 21 | 1014 | 13640 | 11 |
| 316.9 | 1604 | 384 | 327 | 23 | 1024 | 13460 | 14 |
| 315.8 | 1617 | 234 | 298 | 21 | 1008 | 13790 | 11 |
| 316.1 | 1597 | 370 | 324 | 22 | 1000 | 13650 | 11 |
| 317.6 | 1606 | 324 | 314 | 27 | 994 | 13640 | 13 |
| 315.5 | 1604 | 364 | 310 | 23 | 1020 | 13700 | 15 |
| 315.0 | 1615 | 563 | 317 | 26 | 1004 | 14010 | 14 |
| 316.4 | 1618 | 499 | 357 | 25 | 1004 | 14240 | 10 |
| 314.1 | 1582 | 325 | 284 | 16 | 1000 | 13450 | 9 |
| 316.5 | 1601 | 384 | 327 | 24 | 998 | 13950 | 13 |
| 315.4 | 1621 | 304 | 337 | 25 | 1000 | 14030 | 13 |
| 315.1 | 1632 | 310 | 351 | 26 | 998 | 13760 | 14 |
| 315.8 | 1613 | 337 | 309 | 22 | 1516 | 13930 | 13 |
| 316.9 | 1622 | 322 | 353 | 27 | 1000 | 13560 | 15 |
| 313.6 | 1605 | 557 | 331 | 25 | 1008 | 14650 | 12 |
| 314.0 | 1615 | 405 | 344 | 30 | 1000 | 14440 | 14 |
| 313.9 | 1611 | 478 | 324 | 20 | 1000 | 14690 | 11 |
| 314.3 | 1621 | 340 | 322 | 39 | 514 | 15250 | 16 |
| 315.0 | 1605 | 461 | 322 | 36 | 1016 | 14640 | 16 |
| 316.1 | 1614 | 400 | 352 | 34 | 990 | 14400 | 15 |
| 314.4 | 1605 | 449 | 316 | 29 | 534 | 14850 | 8 |
| 314.9 | 1622 | 272 | 345 | 27 | 1010 | 14110 | 15 |
| 315.7 | 1622 | 164 | 291 | 35 | 1034 | 14090 | 20 |
| 315.0 | 1620 | 299 | 322 | 24 | 1016 | 13890 | 12 |
| 313.4 | 1578 | 357 | 301 | 20 | 514 | 14180 | 12 |
| 314.8 | 1620 | 366 | 331 | 25 | 700 | 14440 | 11 |
| 315.9 | 1619 | 451 | 358 | 31 | 894 | 14300 | 14 |
| 313.7 | 1616 | 424 | 325 | 41 | 1008 | 14190 | 21 |
| 313.1 | 1606 | 408 | 303 | 39 | 800 | 14430 | 17 |
| 314.2 | 1604 | 550 | 327 | 31 | 800 | 14330 | 18 |
| 313.3 | 1596 | 529 | 288 | 42 | 1000 | 14660 | 17 |
| 313.5 | 1604 | 457 | 324 | 31 | 510 | 13940 | 17 |
| 316.6 | 1616 | 368 | 347 | 34 | 478 | 13940 | 20 |
| 314.7 | 1611 | 489 | 337 | 37 | 1010 | 14210 | 18 |
| 309.5 | 1627 | 415 | 326 | 36 | 1010 | 13780 | 15 |
| 314.1 | 1642 | 277 | 376 | 39 | 1520 | 13640 | 17 |
| 305.3 | 1652 | 274 | 336 | 28 | 910 | 13310 | 16 |
| 312.9 | 1587 | 288 | 287 | 17 | 1678 | 13930 | 10 |
| 315.8 | 1618 | 428 | 393 | 40 | 1014 | 13620 | 19 |

Table 1.2 : Testing data used for dephosphorization problem
(96 examples)

| Hot Metal Weight (tons) | T (°C) | C % x 100 | Mn % x 100 | Init P % x 100 | Ore (kg) | O ₂ blown (Nm ³) | Final P % x 100 |
|----------------------------------|-----------|-----------------|------------------|-------------------------|-------------|---|--------------------------|
|----------------------------------|-----------|-----------------|------------------|-------------------------|-------------|---|--------------------------|

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 315.8 | 1605 | 409 | 331 | 25 | 1024 | 13860 | 12 |
| 316.0 | 1635 | 353 | 377 | 34 | 1010 | 14180 | 16 |
| 316.3 | 1620 | 294 | 345 | 32 | 520 | 13920 | 18 |
| 314.9 | 1597 | 423 | 342 | 26 | 524 | 14020 | 17 |
| 315.2 | 1617 | 410 | 366 | 35 | 1010 | 13930 | 21 |
| 315.6 | 1642 | 266 | 333 | 29 | 1006 | 13860 | 15 |
| 313.5 | 1608 | 546 | 349 | 38 | 1034 | 14470 | 16 |
| 314.7 | 1623 | 389 | 358 | 30 | 1000 | 13820 | 16 |
| 315.9 | 1603 | 471 | 331 | 34 | 1020 | 14530 | 12 |
| 315.7 | 1614 | 408 | 338 | 25 | 1020 | 14250 | 11 |
| 316.2 | 1613 | 376 | 327 | 29 | 1014 | 14240 | 14 |
| 313.1 | 1597 | 426 | 354 | 32 | 494 | 14780 | 13 |
| 315.1 | 1612 | 465 | 344 | 26 | 1020 | 14280 | 13 |
| 316.5 | 1618 | 494 | 336 | 25 | 1024 | 14440 | 11 |
| 316.4 | 1605 | 516 | 311 | 21 | 1014 | 14620 | 11 |
| 315.5 | 1602 | 478 | 313 | 23 | 804 | 14550 | 10 |
| 315.8 | 1617 | 395 | 320 | 30 | 1008 | 14220 | 12 |
| 315.4 | 1616 | 360 | 336 | 30 | 1000 | 13960 | 17 |
| 315.5 | 1621 | 356 | 320 | 26 | 1030 | 13410 | 15 |
| 315.1 | 1610 | 413 | 304 | 21 | 1000 | 13640 | 12 |
| 313.4 | 1597 | 381 | 307 | 22 | 510 | 14080 | 12 |
| 314.3 | 1580 | 479 | 304 | 21 | 540 | 14030 | 10 |
| 314.9 | 1600 | 415 | 350 | 29 | 514 | 14580 | 14 |
| 315.5 | 1621 | 472 | 356 | 27 | 1020 | 14280 | 15 |
| 314.9 | 1603 | 514 | 326 | 26 | 1000 | 14340 | 11 |
| 317.2 | 1624 | 376 | 349 | 28 | 998 | 14420 | 12 |
| 316.9 | 1618 | 447 | 351 | 26 | 920 | 14220 | 15 |
| 315.3 | 1623 | 369 | 318 | 23 | 520 | 13980 | 14 |
| 315.1 | 1622 | 393 | 349 | 31 | 998 | 14030 | 14 |
| 315.6 | 1669 | 342 | 348 | 25 | 1000 | 14422 | 12 |
| 313.5 | 1622 | 413 | 372 | 27 | 500 | 14520 | 16 |
| 314.6 | 1601 | 562 | 362 | 23 | 1000 | 14310 | 12 |
| 314.5 | 1626 | 491 | 367 | 30 | 514 | 13800 | 19 |
| 314.8 | 1620 | 403 | 350 | 27 | 1004 | 13850 | 16 |
| 313.9 | 1615 | 449 | 362 | 25 | 1020 | 13570 | 17 |
| 313.8 | 1602 | 528 | 343 | 24 | 1040 | 13680 | 13 |
| 315.2 | 1611 | 424 | 362 | 23 | 504 | 13320 | 17 |
| 314.0 | 1597 | 485 | 327 | 23 | 510 | 13840 | 14 |

Table 1.2 continues

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 313.9 | 1628 | 252 | 334 | 27 | 504 | 14010 | 14 |
| 313.0 | 1597 | 460 | 344 | 24 | 400 | 14060 | 14 |
| 315.1 | 1623 | 503 | 376 | 32 | 1000 | 14020 | 16 |
| 314.3 | 1621 | 490 | 337 | 35 | 1020 | 13860 | 18 |
| 315.1 | 1631 | 432 | 352 | 26 | 1004 | 13740 | 15 |
| 314.8 | 1623 | 430 | 344 | 24 | 1010 | 13730 | 13 |
| 315.6 | 1625 | 369 | 364 | 36 | 1014 | 13780 | 17 |
| 315.4 | 1631 | 389 | 351 | 25 | 1014 | 14040 | 13 |
| 314.7 | 1610 | 384 | 332 | 25 | 544 | 13680 | 13 |
| 313.1 | 1612 | 409 | 325 | 23 | 844 | 14240 | 9 |
| 314.8 | 1612 | 288 | 315 | 26 | 508 | 13760 | 14 |
| 315.0 | 1598 | 506 | 330 | 32 | 420 | 14370 | 15 |
| 314.6 | 1619 | 534 | 361 | 36 | 584 | 14110 | 20 |
| 315.2 | 1618 | 503 | 333 | 31 | 1000 | 14520 | 13 |
| 315.7 | 1623 | 300 | 326 | 24 | 500 | 13360 | 15 |
| 314.5 | 1588 | 276 | 276 | 16 | 504 | 13500 | 10 |
| 317.9 | 1591 | 441 | 320 | 27 | 1000 | 13950 | 11 |
| 314.5 | 1603 | 274 | 307 | 15 | 478 | 13440 | 10 |
| 315.1 | 1596 | 497 | 340 | 22 | 530 | 13960 | 11 |
| 316.2 | 1611 | 448 | 331 | 33 | 1004 | 14290 | 17 |
| 317.1 | 1618 | 307 | 343 | 22 | 1004 | 13760 | 13 |
| 315.9 | 1588 | 455 | 304 | 19 | 1000 | 13970 | 10 |
| 316.1 | 1625 | 295 | 371 | 25 | 998 | 13890 | 14 |
| 314.9 | 1611 | 352 | 337 | 20 | 880 | 13500 | 13 |
| 316.5 | 1623 | 355 | 349 | 24 | 1000 | 13650 | 13 |
| 315.9 | 1610 | 431 | 325 | 23 | 1004 | 13670 | 13 |
| 315.4 | 1628 | 269 | 308 | 23 | 594 | 13890 | 14 |
| 315.9 | 1626 | 364 | 351 | 37 | 824 | 13850 | 18 |
| 314.0 | 1633 | 225 | 361 | 23 | 1018 | 13610 | 14 |
| 316.0 | 1598 | 257 | 305 | 19 | 524 | 13810 | 12 |
| 314.7 | 1612 | 304 | 321 | 22 | 1028 | 13460 | 12 |
| 314.8 | 1597 | 547 | 349 | 27 | 830 | 14000 | 13 |
| 315.8 | 1628 | 259 | 356 | 24 | 500 | 13720 | 15 |
| 315.5 | 1622 | 446 | 381 | 24 | 600 | 13870 | 15 |
| 312.0 | 1606 | 302 | 318 | 28 | 1004 | 13870 | 12 |
| 311.2 | 1610 | 393 | 332 | 28 | 1000 | 14220 | 11 |
| 316.0 | 1628 | 347 | 358 | 26 | 864 | 13840 | 14 |
| 317.2 | 1607 | 554 | 349 | 24 | 1478 | 14420 | 12 |
| 315.8 | 1605 | 442 | 337 | 22 | 1004 | 13840 | 12 |
| 315.4 | 1614 | 383 | 335 | 22 | 980 | 13480 | 11 |
| 314.4 | 1640 | 270 | 352 | 24 | 500 | 13670 | 14 |
| 314.5 | 1629 | 364 | 358 | 25 | 510 | 13670 | 13 |
| 314.5 | 1612 | 369 | 357 | 23 | 514 | 14070 | 13 |
| 315.2 | 1591 | 605 | 345 | 30 | 814 | 13840 | 12 |
| 317.4 | 1585 | 469 | 330 | 27 | 500 | 14160 | 15 |
| 315.2 | 1582 | 443 | 345 | 30 | 500 | 14130 | 14 |
| 315.8 | 1602 | 397 | 361 | 29 | 410 | 14130 | 14 |
| 316.3 | 1617 | 552 | 382 | 34 | 1008 | 13930 | 18 |
| 314.2 | 1598 | 323 | 327 | 24 | 1014 | 13870 | 15 |
| 315.4 | 1603 | 423 | 346 | 29 | 1000 | 14480 | 17 |
| 315.5 | 1603 | 411 | 343 | 25 | 1000 | 13960 | 11 |
| 315.3 | 1624 | 278 | 358 | 26 | 1000 | 13870 | 13 |
| 313.7 | 1602 | 462 | 348 | 26 | 1020 | 13790 | 13 |
| 313.9 | 1610 | 421 | 335 | 22 | 1014 | 13780 | 14 |
| | | | | | 1008 | 14110 | 12 |
| | | | | | | 14010 | 11 |

Table 1.2 continues

| | | | | | | | |
|-------|------|-----|-----|----|------|-------|----|
| 314.2 | 1620 | 186 | 301 | 25 | 1014 | 13640 | 12 |
| 316.1 | 1626 | 447 | 360 | 28 | 1014 | 14120 | 14 |
| 314.9 | 1634 | 397 | 372 | 31 | 814 | 13890 | 19 |
| 314.9 | 1590 | 517 | 304 | 20 | 490 | 14410 | 10 |

Table 1.3 : Learning data used for desulphurization problem
(350 examples)

| Initial S (%) | Tempera- ture (°C) | weight pig iron (tons) | weight CaD (kg) | Final S (%) |
|-----------------------|----------------------------|------------------------------|-----------------------|---------------------|
|-----------------------|----------------------------|------------------------------|-----------------------|---------------------|

| | | | | |
|-------|--------|-------|--------|-------|
| 0.075 | 1417.0 | 230.0 | 1408.0 | 0.009 |
| 0.067 | 1369.0 | 196.0 | 1546.0 | 0.006 |
| 0.039 | 1383.0 | 215.0 | 1342.0 | 0.003 |
| 0.060 | 1413.0 | 220.0 | 1661.0 | 0.002 |
| 0.046 | 1435.0 | 240.0 | 1619.0 | 0.001 |
| 0.084 | 1429.0 | 230.0 | 1561.0 | 0.006 |
| 0.085 | 1371.0 | 244.0 | 1665.0 | 0.007 |
| 0.087 | 1382.0 | 215.0 | 1559.0 | 0.006 |
| 0.074 | 1374.0 | 228.0 | 1541.0 | 0.004 |
| 0.070 | 1382.0 | 246.0 | 1622.0 | 0.005 |
| 0.032 | 1400.0 | 180.0 | 1015.0 | 0.003 |
| 0.032 | 1433.0 | 190.0 | 1072.0 | 0.005 |
| 0.032 | 1413.0 | 229.0 | 1292.0 | 0.002 |
| 0.034 | 1436.0 | 230.0 | 1340.0 | 0.003 |
| 0.037 | 1401.0 | 239.0 | 1454.0 | 0.003 |
| 0.031 | 1429.0 | 230.0 | 1275.0 | 0.003 |
| 0.031 | 1439.0 | 220.0 | 1220.0 | 0.004 |
| 0.035 | 1382.0 | 216.0 | 1277.0 | 0.004 |
| 0.095 | 1401.0 | 230.0 | 2000.0 | 0.006 |
| 0.039 | 1468.0 | 231.0 | 1442.0 | 0.006 |
| 0.094 | 1420.0 | 211.0 | 1881.0 | 0.004 |
| 0.054 | 1384.0 | 260.0 | 1880.0 | 0.005 |
| 0.056 | 1397.0 | 205.0 | 1212.0 | 0.006 |
| 0.052 | 1429.0 | 224.0 | 1594.0 | 0.005 |
| 0.069 | 1391.0 | 225.0 | 1795.0 | 0.004 |
| 0.049 | 1386.0 | 230.0 | 1595.0 | 0.005 |
| 0.055 | 1404.0 | 217.0 | 1581.0 | 0.004 |

Table 1.3 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.052 | 1390.0 | 230.0 | 1637.0 | 0.004 |
| 0.048 | 1444.0 | 190.0 | 1306.0 | 0.006 |
| 0.045 | 1396.0 | 236.0 | 1576.0 | 0.002 |
| 0.076 | 1387.0 | 220.0 | 1820.0 | 0.004 |
| 0.074 | 1401.0 | 195.0 | 1597.0 | 0.008 |
| 0.056 | 1412.0 | 190.0 | 1395.0 | 0.007 |
| 0.060 | 1376.0 | 223.0 | 1684.0 | 0.006 |
| 0.066 | 1410.0 | 200.0 | 1568.0 | 0.009 |
| 0.088 | 1405.0 | 251.0 | 2000.0 | 0.003 |
| 0.072 | 1375.0 | 239.0 | 1937.0 | 0.005 |
| 0.115 | 1393.0 | 209.0 | 1992.0 | 0.017 |
| 0.054 | 1424.0 | 240.0 | 1736.0 | 0.002 |
| 0.067 | 1379.0 | 226.0 | 1783.0 | 0.002 |
| 0.056 | 1373.0 | 210.0 | 1425.0 | 0.007 |
| 0.063 | 1423.0 | 230.0 | 1644.0 | 0.003 |
| 0.050 | 1400.0 | 240.0 | 1174.0 | 0.005 |
| 0.050 | 1439.0 | 220.0 | 851.0 | 0.009 |
| 0.058 | 1385.0 | 216.0 | 1609.0 | 0.002 |
| 0.038 | 1394.0 | 198.0 | 1220.0 | 0.004 |
| 0.061 | 1426.0 | 250.0 | 1901.0 | 0.003 |
| 0.068 | 1406.0 | 216.0 | 1713.0 | 0.006 |
| 0.048 | 1400.0 | 231.0 | 1588.0 | 0.002 |
| 0.059 | 1404.0 | 250.0 | 1875.0 | 0.004 |
| 0.059 | 1422.0 | 191.0 | 1433.0 | 0.008 |
| 0.049 | 1421.0 | 210.0 | 1157.0 | 0.008 |
| 0.066 | 1385.0 | 256.0 | 1642.0 | 0.006 |
| 0.058 | 1409.0 | 220.0 | 1324.0 | 0.008 |
| 0.059 | 1401.0 | 187.0 | 1135.0 | 0.008 |
| 0.064 | 1385.0 | 237.0 | 1498.0 | 0.004 |
| 0.077 | 1393.0 | 200.0 | 1376.0 | 0.009 |
| 0.07 | 1415.0 | 190.0 | 1405.0 | 0.004 |
| 0.068 | 1381.0 | 228.0 | 1809.0 | 0.005 |
| 0.076 | 1374.0 | 210.0 | 1737.0 | 0.005 |
| 0.096 | 1383.0 | 203.0 | 1823.0 | 0.005 |
| 0.071 | 1419.0 | 240.0 | 1935.0 | 0.003 |
| 0.041 | 1436.0 | 190.0 | 1215.0 | 0.005 |
| 0.061 | 1386.0 | 182.0 | 1384.0 | 0.007 |
| 0.044 | 1410.0 | 240.0 | 1586.0 | 0.004 |
| 0.035 | 1420.0 | 200.0 | 1183.0 | 0.005 |
| 0.062 | 1368.0 | 231.0 | 1640.0 | 0.007 |
| 0.039 | 1393.0 | 220.0 | 1251.0 | 0.009 |
| 0.056 | 1399.0 | 221.0 | 1500.0 | 0.004 |
| 0.064 | 1444.0 | 240.0 | 1727.0 | 0.003 |
| 0.057 | 1396.0 | 199.0 | 1472.0 | 0.008 |
| 0.039 | 1398.0 | 245.0 | 1529.0 | 0.002 |
| 0.044 | 1431.0 | 240.0 | 1586.0 | 0.003 |
| 0.036 | 1421.0 | 200.0 | 1089.0 | 0.007 |
| 0.051 | 1428.0 | 260.0 | 1691.0 | 0.005 |
| 0.074 | 1402.0 | 230.0 | 1399.0 | 0.008 |
| 0.063 | 1401.0 | 192.0 | 1478.0 | 0.006 |
| 0.088 | 1369.0 | 249.0 | 2000.0 | 0.007 |
| 0.090 | 1367.0 | 180.0 | 1581.0 | 0.015 |
| 0.100 | 1352.0 | 208.0 | 1894.0 | 0.008 |
| 0.076 | 1380.0 | 250.0 | 2000.0 | 0.007 |

Table 1.3 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.069 | 1402.0 | 220.0 | 1755.0 | 0.069 |
| 0.076 | 1383.0 | 245.0 | 2000.0 | 0.007 |
| 0.089 | 1368.0 | 213.0 | 1864.0 | 0.006 |
| 0.071 | 1353.0 | 210.0 | 1693.0 | 0.007 |
| 0.064 | 1379.0 | 223.0 | 1409.0 | 0.006 |
| 0.076 | 1424.0 | 200.0 | 1368.0 | 0.010 |
| 0.060 | 1425.0 | 210.0 | 1286.0 | 0.009 |
| 0.036 | 1380.0 | 209.0 | 1254.0 | 0.004 |
| 0.125 | 1400.0 | 190.0 | 1859.0 | 0.007 |
| 0.039 | 1375.0 | 213.0 | 1330.0 | 0.007 |
| 0.066 | 1368.0 | 246.0 | 1929.0 | 0.005 |
| 0.045 | 1387.0 | 206.0 | 1376.0 | 0.008 |
| 0.044 | 1390.0 | 230.0 | 1520.0 | 0.005 |
| 0.050 | 1374.0 | 177.0 | 1238.0 | 0.011 |
| 0.046 | 1408.0 | 230.0 | 1551.0 | 0.005 |
| 0.041 | 1423.0 | 260.0 | 1663.0 | 0.003 |
| 0.064 | 1389.0 | 183.0 | 1156.0 | 0.007 |
| 0.050 | 1396.0 | 207.0 | 1012.0 | 0.008 |
| 0.075 | 1392.0 | 247.0 | 1680.0 | 0.009 |
| 0.071 | 1397.0 | 230.0 | 1855.0 | 0.004 |
| 0.058 | 1412.0 | 180.0 | 1084.0 | 0.006 |
| 0.060 | 1401.0 | 220.0 | 1661.0 | 0.006 |
| 0.065 | 1395.0 | 205.0 | 1598.0 | 0.006 |
| 0.059 | 1400.0 | 233.0 | 1748.0 | 0.006 |
| 0.041 | 1450.0 | 210.0 | 1343.0 | 0.005 |
| 0.066 | 1392.0 | 216.0 | 1694.0 | 0.004 |
| 0.035 | 1449.0 | 240.0 | 1419.0 | 0.003 |
| 0.049 | 1400.0 | 229.0 | 1588.0 | 0.003 |
| 0.062 | 1391.0 | 222.0 | 1699.0 | 0.005 |
| 0.085 | 1381.0 | 227.0 | 1955.0 | 0.004 |
| 0.057 | 1370.0 | 193.0 | 1427.0 | 0.005 |
| 0.075 | 1387.0 | 259.0 | 2000.0 | 0.008 |
| 0.025 | 1409.0 | 250.0 | 1223.0 | 0.004 |
| 0.075 | 1389.0 | 208.0 | 1712.0 | 0.006 |
| 0.037 | 1462.0 | 250.0 | 1521.0 | 0.002 |
| 0.023 | 1450.0 | 240.0 | 1113.0 | 0.002 |
| 0.078 | 1387.0 | 259.0 | 2000.0 | 0.004 |
| 0.049 | 1406.0 | 202.0 | 1401.0 | 0.005 |
| 0.030 | 1415.0 | 220.0 | 1198.0 | 0.002 |
| 0.029 | 1421.0 | 190.0 | 1015.0 | 0.004 |
| 0.024 | 1386.0 | 235.0 | 1120.0 | 0.003 |
| 0.025 | 1431.0 | 225.0 | 1100.0 | 0.002 |
| 0.079 | 1385.0 | 224.0 | 1559.0 | 0.005 |
| 0.064 | 1400.0 | 255.0 | 1612.0 | 0.005 |
| 0.064 | 1407.0 | 203.0 | 1283.0 | 0.006 |
| 0.056 | 1387.0 | 234.0 | 1718.0 | 0.002 |
| 0.037 | 1446.0 | 210.0 | 1277.0 | 0.003 |
| 0.062 | 1369.0 | 205.0 | 1569.0 | 0.006 |
| 0.024 | 1418.0 | 270.0 | 1287.0 | 0.002 |
| 0.037 | 1389.0 | 240.0 | 1031.0 | 0.003 |
| 0.065 | 1383.0 | 219.0 | 1394.0 | 0.006 |
| 0.054 | 1400.0 | 258.0 | 1405.0 | 0.004 |
| 0.032 | 1405.0 | 240.0 | 925.0 | 0.003 |
| 0.052 | 1397.0 | 226.0 | 1608.0 | 0.002 |

Table 1.3 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.039 | 1401.0 | 210.0 | 1311.0 | 0.003 |
| 0.082 | 1405.0 | 240.0 | 2000.0 | 0.004 |
| 0.053 | 1416.0 | 202.0 | 1449.0 | 0.002 |
| 0.044 | 1388.0 | 207.0 | 1368.0 | 0.002 |
| 0.037 | 1422.0 | 166.0 | 660.0 | 0.006 |
| 0.058 | 1376.0 | 240.0 | 1445.0 | 0.008 |
| 0.049 | 1389.0 | 210.0 | 799.0 | 0.012 |
| 0.045 | 1401.0 | 240.0 | 851.0 | 0.009 |
| 0.043 | 1418.0 | 240.0 | 818.0 | 0.009 |
| 0.057 | 1369.0 | 240.0 | 1136.0 | 0.009 |
| 0.054 | 1401.0 | 170.0 | 871.0 | 0.009 |
| 0.054 | 1428.0 | 240.0 | 934.0 | 0.010 |
| 0.059 | 1410.0 | 230.0 | 1725.0 | 0.005 |
| 0.041 | 1397.0 | 218.0 | 1394.0 | 0.005 |
| 0.060 | 1435.0 | 210.0 | 1211.0 | 0.006 |
| 0.056 | 1384.0 | 195.0 | 1432.0 | 0.003 |
| 0.050 | 1409.0 | 245.0 | 1714.0 | 0.004 |
| 0.041 | 1431.0 | 190.0 | 943.0 | 0.005 |
| 0.054 | 1375.0 | 244.0 | 1765.0 | 0.006 |
| 0.064 | 1408.0 | 200.0 | 1550.0 | 0.007 |
| 0.077 | 1389.0 | 180.0 | 1239.0 | 0.007 |
| 0.053 | 1400.0 | 218.0 | 1564.0 | 0.004 |
| 0.067 | 1356.0 | 221.0 | 1743.0 | 0.004 |
| 0.056 | 1387.0 | 211.0 | 1248.0 | 0.006 |
| 0.065 | 1399.0 | 237.0 | 1509.0 | 0.006 |
| 0.046 | 1400.0 | 225.0 | 1196.0 | 0.005 |
| 0.063 | 1375.0 | 180.0 | 1386.0 | 0.006 |
| 0.058 | 1418.0 | 223.0 | 1661.0 | 0.006 |
| 0.048 | 1360.0 | 194.0 | 1333.0 | 0.006 |
| 0.066 | 1368.0 | 235.0 | 1843.0 | 0.006 |
| 0.058 | 1372.0 | 244.0 | 1818.0 | 0.005 |
| 0.056 | 1384.0 | 242.0 | 1777.0 | 0.005 |
| 0.062 | 1384.0 | 235.0 | 1798.0 | 0.004 |
| 0.060 | 1397.0 | 240.0 | 1813.0 | 0.003 |
| 0.067 | 1367.0 | 215.0 | 1696.0 | 0.004 |
| 0.048 | 1419.0 | 220.0 | 1512.0 | 0.003 |
| 0.063 | 1379.0 | 190.0 | 1463.0 | 0.003 |
| 0.045 | 1426.0 | 260.0 | 1736.0 | 0.002 |
| 0.044 | 1425.0 | 190.0 | 1256.0 | 0.006 |
| 0.060 | 1416.0 | 220.0 | 1661.0 | 0.004 |
| 0.041 | 1407.0 | 211.0 | 1349.0 | 0.003 |
| 0.057 | 1377.0 | 185.0 | 1368.0 | 0.003 |
| 0.069 | 1425.0 | 220.0 | 1291.0 | 0.005 |
| 0.050 | 1386.0 | 179.0 | 875.0 | 0.006 |
| 0.054 | 1396.0 | 220.0 | 1591.0 | 0.004 |
| 0.092 | 1422.0 | 200.0 | 1770.0 | 0.005 |
| 0.075 | 1413.0 | 230.0 | 1893.0 | 0.002 |
| 0.059 | 1390.0 | 231.0 | 1733.0 | 0.006 |
| 0.057 | 1420.0 | 240.0 | 1642.0 | 0.004 |
| 0.041 | 1401.0 | 250.0 | 1599.0 | 0.002 |
| 0.043 | 1388.0 | 220.0 | 1439.0 | 0.005 |
| 0.053 | 1380.0 | 217.0 | 1557.0 | 0.003 |
| 0.046 | 1414.0 | 251.0 | 1693.0 | 0.001 |
| 0.032 | 1414.0 | 200.0 | 1017.0 | 0.005 |

Table 1.3 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.042 | 1408.0 | 200.0 | 1183.0 | 0.006 |
| 0.074 | 1386.0 | 258.0 | 1970.0 | 0.005 |
| 0.050 | 1399.0 | 217.0 | 1398.0 | 0.004 |
| 0.030 | 1388.0 | 185.0 | 905.0 | 0.004 |
| 0.066 | 1374.0 | 214.0 | 1372.0 | 0.004 |
| 0.054 | 1392.0 | 176.0 | 1021.0 | 0.006 |
| 0.072 | 1361.0 | 235.0 | 1800.0 | 0.004 |
| 0.068 | 1377.0 | 250.0 | 1800.0 | 0.005 |
| 0.073 | 1396.0 | 230.0 | 1800.0 | 0.006 |
| 0.057 | 1398.0 | 240.0 | 1432.0 | 0.004 |
| 0.080 | 1424.0 | 230.0 | 1800.0 | 0.005 |
| 0.075 | 1411.0 | 250.0 | 1800.0 | 0.004 |
| 0.077 | 1367.0 | 239.0 | 1800.0 | 0.005 |
| 0.063 | 1405.0 | 265.0 | 1800.0 | 0.003 |
| 0.066 | 1394.0 | 230.0 | 1800.0 | 0.005 |
| 0.080 | 1379.0 | 240.0 | 1800.0 | 0.008 |
| 0.058 | 1389.0 | 265.0 | 1800.0 | 0.004 |
| 0.055 | 1419.0 | 250.0 | 1800.0 | 0.003 |
| 0.077 | 1386.0 | 212.0 | 1762.0 | 0.011 |
| 0.075 | 1367.0 | 223.0 | 1800.0 | 0.012 |
| 0.037 | 1412.0 | 250.0 | 1521.0 | 0.002 |
| 0.035 | 1406.0 | 219.0 | 1295.0 | 0.011 |
| 0.060 | 1379.0 | 205.0 | 1255.0 | 0.002 |
| 0.053 | 1385.0 | 185.0 | 1063.0 | 0.001 |
| 0.065 | 1377.0 | 210.0 | 1337.0 | 0.006 |
| 0.057 | 1366.0 | 245.0 | 1462.0 | 0.001 |
| 0.054 | 1392.0 | 222.0 | 1605.0 | 0.002 |
| 0.039 | 1398.0 | 240.0 | 1498.0 | 0.002 |
| 0.032 | 1437.0 | 200.0 | 1128.0 | 0.001 |
| 0.034 | 1404.0 | 250.0 | 1456.0 | 0.002 |
| 0.059 | 1375.0 | 203.0 | 1095.0 | 0.008 |
| 0.045 | 1386.0 | 260.0 | 1736.0 | 0.002 |
| 0.033 | 1411.0 | 235.0 | 1348.0 | 0.002 |
| 0.038 | 1400.0 | 197.0 | 1214.0 | 0.004 |
| 0.064 | 1372.0 | 241.0 | 1734.0 | 0.002 |
| 0.044 | 1405.0 | 240.0 | 1586.0 | 0.002 |
| 0.043 | 1429.0 | 261.0 | 1707.0 | 0.001 |
| 0.058 | 1379.0 | 218.0 | 1624.0 | 0.004 |
| 0.028 | 1410.0 | 250.0 | 1309.0 | 0.002 |
| 0.033 | 1415.0 | 235.0 | 1348.0 | 0.002 |
| 0.023 | 1447.0 | 205.0 | 951.0 | 0.002 |
| 0.034 | 1445.0 | 230.0 | 929.0 | 0.003 |
| 0.036 | 1405.0 | 210.0 | 1260.0 | 0.006 |
| 0.043 | 1445.0 | 240.0 | 1569.0 | 0.001 |
| 0.045 | 1386.0 | 232.0 | 1218.0 | 0.004 |
| 0.033 | 1429.0 | 220.0 | 798.0 | 0.006 |
| 0.060 | 1382.0 | 233.0 | 1760.0 | 0.004 |
| 0.033 | 1394.0 | 200.0 | 1147.0 | 0.002 |
| 0.031 | 1441.0 | 240.0 | 988.0 | 0.007 |
| 0.025 | 1418.0 | 221.0 | 765.0 | 0.002 |
| 0.033 | 1447.0 | 210.0 | 1204.0 | 0.002 |
| 0.030 | 1402.0 | 230.0 | 1252.0 | 0.003 |
| 0.035 | 1433.0 | 260.0 | 1538.0 | 0.002 |
| 0.028 | 1417.0 | 250.0 | 782.0 | 0.012 |

Table 1.3 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.061 | 1380.0 | 199.0 | 1228.0 | 0.002 |
| 0.048 | 1416.0 | 230.0 | 1252.0 | 0.006 |
| 0.060 | 1377.0 | 236.0 | 1445.0 | 0.004 |
| 0.052 | 1425.0 | 250.0 | 1422.0 | 0.004 |
| 0.036 | 1427.0 | 260.0 | 1012.0 | 0.001 |
| 0.028 | 1371.0 | 220.0 | 688.0 | 0.003 |
| 0.037 | 1416.0 | 203.0 | 1235.0 | 0.001 |
| 0.067 | 1367.0 | 212.0 | 1225.0 | 0.007 |
| 0.047 | 1378.0 | 247.0 | 1162.0 | 0.006 |
| 0.049 | 1375.0 | 200.0 | 1387.0 | 0.008 |
| 0.048 | 1390.0 | 222.0 | 1526.0 | 0.003 |
| 0.044 | 1367.0 | 220.0 | 1454.0 | 0.006 |
| 0.054 | 1407.0 | 220.0 | 1591.0 | 0.002 |
| 0.060 | 1386.0 | 223.0 | 1684.0 | 0.002 |
| 0.057 | 1410.0 | 220.0 | 1627.0 | 0.004 |
| 0.054 | 1396.0 | 218.0 | 1577.0 | 0.005 |
| 0.040 | 1399.0 | 154.0 | 973.0 | 0.010 |
| 0.033 | 1396.0 | 250.0 | 1434.0 | 0.003 |
| 0.044 | 1389.0 | 197.0 | 1302.0 | 0.003 |
| 0.035 | 1395.0 | 218.0 | 1289.0 | 0.001 |
| 0.067 | 1382.0 | 227.0 | 1790.0 | 0.003 |
| 0.058 | 1396.0 | 230.0 | 1302.0 | 0.004 |
| 0.066 | 1372.0 | 200.0 | 1211.0 | 0.011 |
| 0.060 | 1399.0 | 220.0 | 1268.0 | 0.005 |
| 0.048 | 1377.0 | 180.0 | 858.0 | 0.011 |
| 0.047 | 1405.0 | 220.0 | 1184.0 | 0.004 |
| 0.048 | 1370.0 | 211.0 | 1149.0 | 0.007 |
| 0.028 | 1358.0 | 190.0 | 995.0 | 0.002 |
| 0.046 | 1383.0 | 203.0 | 1079.0 | 0.006 |
| 0.039 | 1395.0 | 185.0 | 765.0 | 0.004 |
| 0.052 | 1384.0 | 194.0 | 972.0 | 0.003 |
| 0.045 | 1404.0 | 240.0 | 1097.0 | 0.006 |
| 0.043 | 1402.0 | 200.0 | 886.0 | 0.007 |
| 0.032 | 1437.0 | 173.0 | 611.0 | 0.004 |
| 0.067 | 1386.0 | 229.0 | 1324.0 | 0.007 |
| 0.081 | 1380.0 | 208.0 | 1322.0 | 0.013 |
| 0.039 | 1382.0 | 230.0 | 951.0 | 0.006 |
| 0.062 | 1377.0 | 191.0 | 1461.0 | 0.002 |
| 0.053 | 1415.0 | 180.0 | 912.0 | 0.005 |
| 0.041 | 1428.0 | 228.0 | 978.0 | 0.004 |
| 0.055 | 1380.0 | 200.0 | 1036.0 | 0.007 |
| 0.051 | 1378.0 | 177.0 | 1249.0 | 0.002 |
| 0.045 | 1387.0 | 240.0 | 1603.0 | 0.004 |
| 0.075 | 1375.0 | 202.0 | 1662.0 | 0.003 |
| 0.042 | 1417.0 | 226.0 | 1462.0 | 0.002 |
| 0.028 | 1407.0 | 253.0 | 1325.0 | 0.002 |
| 0.087 | 1374.0 | 211.0 | 1832.0 | 0.005 |
| 0.058 | 1378.0 | 178.0 | 1326.0 | 0.002 |
| 0.046 | 1418.0 | 200.0 | 1349.0 | 0.002 |
| 0.038 | 1427.0 | 230.0 | 1418.0 | 0.001 |
| 0.035 | 1422.0 | 210.0 | 942.0 | 0.003 |
| 0.045 | 1390.0 | 200.0 | 1050.0 | 0.006 |
| 0.024 | 1446.0 | 205.0 | 684.0 | 0.002 |
| 0.100 | 1365.0 | 234.0 | 1796.0 | 0.004 |

Table 1.3 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.033 | 1439.0 | 200.0 | 861.0 | 0.006 |
| 0.027 | 1465.0 | 210.0 | 776.0 | 0.003 |
| 0.056 | 1359.0 | 221.0 | 1228.0 | 0.003 |
| 0.051 | 1401.0 | 230.0 | 1139.0 | 0.006 |
| 0.049 | 1391.0 | 197.0 | 1014.0 | 0.004 |
| 0.052 | 1378.0 | 210.0 | 1119.0 | 0.003 |
| 0.046 | 1405.0 | 230.0 | 1140.0 | 0.001 |
| 0.035 | 1403.0 | 180.0 | 807.0 | 0.007 |
| 0.053 | 1382.0 | 170.0 | 977.0 | 0.003 |
| 0.044 | 1410.0 | 170.0 | 1123.0 | 0.002 |
| 0.054 | 1400.0 | 240.0 | 1736.0 | 0.002 |
| 0.067 | 1380.0 | 207.0 | 1633.0 | 0.004 |
| 0.048 | 1418.0 | 260.0 | 1787.0 | 0.002 |
| 0.054 | 1356.0 | 150.0 | 1085.0 | 0.005 |
| 0.076 | 1378.0 | 199.0 | 1361.0 | 0.007 |
| 0.048 | 1409.0 | 260.0 | 1416.0 | 0.003 |
| 0.045 | 1403.0 | 210.0 | 960.0 | 0.005 |
| 0.027 | 1436.0 | 210.0 | 634.0 | 0.006 |
| 0.040 | 1414.0 | 168.0 | 708.0 | 0.002 |
| 0.064 | 1397.0 | 193.0 | 1495.0 | 0.007 |
| 0.031 | 1410.0 | 225.0 | 1248.0 | 0.003 |
| 0.036 | 1412.0 | 220.0 | 1320.0 | 0.006 |
| 0.058 | 1355.0 | 179.0 | 1333.0 | 0.003 |
| 0.038 | 1410.0 | 206.0 | 1270.0 | 0.002 |
| 0.072 | 1370.0 | 169.0 | 1370.0 | 0.002 |
| 0.041 | 1353.0 | 210.0 | 1343.0 | 0.004 |
| 0.043 | 1415.0 | 240.0 | 1569.0 | 0.043 |
| 0.046 | 1373.0 | 204.0 | 1376.0 | 0.005 |
| 0.034 | 1385.0 | 187.0 | 1089.0 | 0.004 |
| 0.044 | 1375.0 | 173.0 | 896.0 | 0.004 |
| 0.025 | 1401.0 | 178.0 | 616.0 | 0.002 |
| 0.053 | 1398.0 | 220.0 | 1115.0 | 0.005 |
| 0.030 | 1423.0 | 160.0 | 534.0 | 0.005 |
| 0.025 | 1451.0 | 170.0 | 473.0 | 0.005 |
| 0.020 | 1446.0 | 203.0 | 855.0 | 0.003 |
| 0.062 | 1379.0 | 194.0 | 1484.0 | 0.003 |
| 0.062 | 1369.0 | 214.0 | 1255.0 | 0.007 |
| 0.020 | 1421.0 | 215.0 | 521.0 | 0.002 |
| 0.063 | 1364.0 | 195.0 | 1153.0 | 0.009 |
| 0.025 | 1443.0 | 220.0 | 683.0 | 0.004 |
| 0.026 | 1415.0 | 218.0 | 633.0 | 0.006 |
| 0.086 | 1379.0 | 222.0 | 1452.0 | 0.010 |
| 0.028 | 1413.0 | 198.0 | 619.0 | 0.004 |
| 0.023 | 1440.0 | 220.0 | 1020.0 | 0.001 |
| 0.061 | 1402.0 | 231.0 | 1756.0 | 0.002 |
| 0.017 | 1444.0 | 200.0 | 744.0 | 0.006 |
| 0.022 | 1422.0 | 200.0 | 900.0 | 0.002 |
| 0.022 | 1434.0 | 220.0 | 990.0 | 0.003 |
| 0.052 | 1400.0 | 196.0 | 1395.0 | 0.002 |
| 0.026 | 1343.0 | 230.0 | 1152.0 | 0.003 |
| 0.032 | 1490.0 | 203.0 | 1145.0 | 0.001 |
| 0.021 | 1401.0 | 170.0 | 383.0 | 0.013 |
| 0.014 | 1447.0 | 210.0 | 657.0 | 0.002 |

Table 1.4 : Testing data used for desulphurization problem
(161 examples)

| Initial S (%) | Tempera- ture (°C) | weight pig iron (tons) | weight CaD (kg) | Final S (%) |
|-----------------------|----------------------------|------------------------------|-----------------------|---------------------|
| 0.029 | 1435.0 | 200.0 | 1068.0 | 0.003 |
| 0.022 | 1405.0 | 160.0 | 720.0 | 0.003 |
| 0.043 | 1405.0 | 234.0 | 1530.0 | 0.006 |
| 0.041 | 1408.0 | 172.0 | 1004.0 | 0.001 |
| 0.040 | 1402.0 | 211.0 | 1216.0 | 0.002 |
| 0.041 | 1387.0 | 218.0 | 1273.0 | 0.002 |
| 0.043 | 1388.0 | 242.0 | 1150.0 | 0.003 |
| 0.023 | 1415.0 | 220.0 | 1020.0 | 0.003 |
| 0.025 | 1456.0 | 170.0 | 831.0 | 0.002 |
| 0.042 | 1398.0 | 201.0 | 1300.0 | 0.002 |
| 0.028 | 1418.0 | 220.0 | 1152.0 | 0.005 |
| 0.054 | 1412.0 | 225.0 | 1627.0 | 0.003 |
| 0.037 | 1424.0 | 220.0 | 1338.0 | 0.003 |
| 0.054 | 1387.0 | 250.0 | 1808.0 | 0.001 |
| 0.038 | 1456.0 | 230.0 | 1418.0 | 0.001 |
| 0.042 | 1395.0 | 210.0 | 1358.0 | 0.002 |
| 0.032 | 1436.0 | 215.0 | 1213.0 | 0.002 |
| 0.044 | 1402.0 | 245.0 | 1619.0 | 0.001 |
| 0.030 | 1398.0 | 230.0 | 1252.0 | 0.001 |
| 0.040 | 1408.0 | 211.0 | 1333.0 | 0.002 |
| 0.037 | 1435.0 | 219.0 | 1332.0 | 0.002 |
| 0.044 | 1405.0 | 215.0 | 1421.0 | 0.001 |
| 0.059 | 1394.0 | 222.0 | 1348.0 | 0.004 |
| 0.048 | 1407.0 | 204.0 | 1111.0 | 0.005 |
| 0.028 | 1362.0 | 215.0 | 741.0 | 0.003 |
| 0.026 | 1415.0 | 180.0 | 580.0 | 0.004 |
| 0.087 | 1383.0 | 237.0 | 1719.0 | 0.002 |
| 0.084 | 1372.0 | 193.0 | 1655.0 | 0.005 |
| 0.090 | 1396.0 | 227.0 | 1994.0 | 0.008 |
| 0.032 | 1425.0 | 220.0 | 1241.0 | 0.005 |
| 0.055 | 1372.0 | 228.0 | 1662.0 | 0.002 |
| 0.055 | 1370.0 | 213.0 | 1248.0 | 0.005 |
| 0.043 | 1395.0 | 216.0 | 1104.0 | 0.003 |
| 0.039 | 1393.0 | 231.0 | 1442.0 | 0.002 |
| 0.042 | 1373.0 | 203.0 | 1313.0 | 0.006 |
| 0.029 | 1379.0 | 240.0 | 1282.0 | 0.002 |
| 0.041 | 1382.0 | 202.0 | 1292.0 | 0.004 |
| 0.051 | 1380.0 | 198.0 | 1397.0 | 0.003 |

Table 1.4 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.024 | 1455.0 | 190.0 | 906.0 | 0.003 |
| 0.063 | 1384.0 | 217.0 | 1214.0 | 0.007 |
| 0.024 | 1411.0 | 238.0 | 1134.0 | 0.002 |
| 0.057 | 1357.0 | 172.0 | 1272.0 | 0.001 |
| 0.055 | 1386.0 | 218.0 | 1589.0 | 0.006 |
| 0.029 | 1396.0 | 228.0 | 1218.0 | 0.003 |
| 0.029 | 1428.0 | 220.0 | 712.0 | 0.004 |
| 0.062 | 1399.0 | 229.0 | 1752.0 | 0.004 |
| 0.023 | 1423.0 | 234.0 | 1085.0 | 0.001 |
| 0.021 | 1456.0 | 210.0 | 916.0 | 0.005 |
| 0.021 | 1350.0 | 230.0 | 1003.0 | 0.007 |
| 0.055 | 1391.0 | 215.0 | 1567.0 | 0.004 |
| 0.020 | 1439.0 | 160.0 | 674.0 | 0.006 |
| 0.022 | 1435.0 | 224.0 | 1008.0 | 0.004 |
| 0.064 | 1387.0 | 210.0 | 1627.0 | 0.006 |
| 0.049 | 1382.0 | 178.0 | 1235.0 | 0.005 |
| 0.057 | 1406.0 | 168.0 | 1242.0 | 0.003 |
| 0.049 | 1389.0 | 223.0 | 1228.0 | 0.006 |
| 0.057 | 1380.0 | 212.0 | 1568.0 | 0.004 |
| 0.061 | 1395.0 | 215.0 | 1634.0 | 0.002 |
| 0.074 | 1376.0 | 226.0 | 1851.0 | 0.006 |
| 0.068 | 1365.0 | 156.0 | 1237.0 | 0.003 |
| 0.033 | 1345.0 | 200.0 | 1147.0 | 0.002 |
| 0.043 | 1381.0 | 230.0 | 1175.0 | 0.007 |
| 0.035 | 1401.0 | 180.0 | 1064.0 | 0.004 |
| 0.036 | 1384.0 | 213.0 | 973.0 | 0.010 |
| 0.055 | 1358.0 | 233.0 | 969.0 | 0.010 |
| 0.026 | 1444.0 | 203.0 | 1017.0 | 0.001 |
| 0.035 | 1405.0 | 202.0 | 1194.0 | 0.001 |
| 0.031 | 1436.0 | 229.0 | 1270.0 | 0.005 |
| 0.075 | 1388.0 | 180.0 | 1481.0 | 0.001 |
| 0.034 | 1390.0 | 215.0 | 1252.0 | 0.001 |
| 0.067 | 1380.0 | 207.0 | 1633.0 | 0.003 |
| 0.021 | 1409.0 | 180.0 | 463.0 | 0.002 |
| 0.031 | 1403.0 | 227.0 | 780.0 | 0.003 |
| 0.077 | 1372.0 | 222.0 | 1377.0 | 0.007 |
| 0.031 | 1407.0 | 192.0 | 1064.0 | 0.001 |
| 0.051 | 1392.0 | 201.0 | 1419.0 | 0.002 |
| 0.048 | 1401.0 | 252.0 | 1732.0 | 0.003 |
| 0.033 | 1399.0 | 221.0 | 1267.0 | 0.003 |
| 0.052 | 1379.0 | 200.0 | 1002.0 | 0.007 |
| 0.063 | 1375.0 | 224.0 | 1253.0 | 0.005 |
| 0.025 | 1348.0 | 230.0 | 997.0 | 0.004 |
| 0.075 | 1385.0 | 215.0 | 1770.0 | 0.002 |
| 0.032 | 1419.0 | 250.0 | 1410.0 | 0.002 |
| 0.033 | 1378.0 | 190.0 | 1089.0 | 0.001 |
| 0.086 | 1359.0 | 224.0 | 1937.0 | 0.004 |
| 0.040 | 1391.0 | 250.0 | 1580.0 | 0.003 |
| 0.081 | 1387.0 | 246.0 | 1731.0 | 0.007 |
| 0.069 | 1379.0 | 222.0 | 1771.0 | 0.001 |
| 0.015 | 1424.0 | 199.0 | 664.0 | 0.003 |
| 0.023 | 1425.0 | 230.0 | 738.0 | 0.004 |
| 0.019 | 1459.0 | 230.0 | 933.0 | 0.002 |
| 0.019 | 1433.0 | 215.0 | 872.0 | 0.001 |

Table 1.4 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.024 | 1432.0 | 230.0 | 612.0 | 0.003 |
| 0.022 | 1417.0 | 210.0 | 645.0 | 0.002 |
| 0.032 | 1394.0 | 230.0 | 1297.0 | 0.003 |
| 0.034 | 1379.0 | 170.0 | 990.0 | 0.001 |
| 0.082 | 1373.0 | 220.0 | 1870.0 | 0.001 |
| 0.031 | 1412.0 | 180.0 | 998.0 | 0.002 |
| 0.027 | 1432.0 | 240.0 | 801.0 | 0.001 |
| 0.080 | 1377.0 | 226.0 | 1428.0 | 0.009 |
| 0.081 | 1379.0 | 239.0 | 1682.0 | 0.005 |
| 0.020 | 1440.0 | 189.0 | 526.0 | 0.003 |
| 0.075 | 1376.0 | 204.0 | 1387.0 | 0.009 |
| 0.022 | 1466.0 | 180.0 | 553.0 | 0.005 |
| 0.019 | 1424.0 | 236.0 | 620.0 | 0.002 |
| 0.028 | 1410.0 | 215.0 | 1126.0 | 0.001 |
| 0.033 | 1438.0 | 195.0 | 1118.0 | 0.003 |
| 0.041 | 1399.0 | 225.0 | 1439.0 | 0.002 |
| 0.082 | 1375.0 | 213.0 | 1811.0 | 0.003 |
| 0.088 | 1375.0 | 194.0 | 1691.0 | 0.011 |
| 0.036 | 1416.0 | 210.0 | 1260.0 | 0.003 |
| 0.056 | 1380.0 | 209.0 | 1535.0 | 0.001 |
| 0.039 | 1401.0 | 170.0 | 818.0 | 0.007 |
| 0.060 | 1377.0 | 236.0 | 1445.0 | 0.003 |
| 0.030 | 1417.0 | 220.0 | 805.0 | 0.005 |
| 0.022 | 1434.0 | 210.0 | 570.0 | 0.005 |
| 0.024 | 1382.0 | 216.0 | 721.0 | 0.004 |
| 0.080 | 1396.0 | 205.0 | 1295.0 | 0.007 |
| 0.024 | 1408.0 | 217.0 | 577.0 | 0.009 |
| 0.022 | 1442.0 | 203.0 | 551.0 | 0.007 |
| 0.064 | 1377.0 | 228.0 | 1359.0 | 0.007 |
| 0.017 | 1434.0 | 210.0 | 406.0 | 0.006 |
| 0.070 | 1377.0 | 223.0 | 1470.0 | 0.008 |
| 0.018 | 1402.0 | 220.0 | 856.0 | 0.002 |
| 0.062 | 1373.0 | 232.0 | 1775.0 | 0.006 |
| 0.025 | 1425.0 | 165.0 | 571.0 | 0.004 |
| 0.040 | 1387.0 | 186.0 | 910.0 | 0.003 |
| 0.059 | 1388.0 | 208.0 | 1122.0 | 0.005 |
| 0.020 | 1373.0 | 210.0 | 584.0 | 0.004 |
| 0.050 | 1370.0 | 210.0 | 1027.0 | 0.007 |
| 0.073 | 1391.0 | 218.0 | 1465.0 | 0.010 |
| 0.020 | 1456.0 | 230.0 | 640.0 | 0.003 |
| 0.047 | 1418.0 | 206.0 | 969.0 | 0.007 |
| 0.047 | 1403.0 | 210.0 | 1430.0 | 0.002 |
| 0.055 | 1370.0 | 260.0 | 1895.0 | 0.001 |
| 0.048 | 1405.0 | 230.0 | 1581.0 | 0.005 |
| 0.035 | 1355.0 | 252.0 | 1490.0 | 0.002 |
| 0.029 | 1448.0 | 230.0 | 1229.0 | 0.002 |
| 0.083 | 1374.0 | 209.0 | 1785.0 | 0.005 |
| 0.068 | 1394.0 | 200.0 | 1165.0 | 0.005 |
| 0.044 | 1403.0 | 217.0 | 1124.0 | 0.005 |
| 0.089 | 1388.0 | 222.0 | 1943.0 | 0.030 |
| 0.120 | 1386.0 | 176.0 | 1700.0 | 0.018 |
| 0.021 | 1388.0 | 200.0 | 872.0 | 0.007 |
| 0.021 | 1466.0 | 230.0 | 1003.0 | 0.003 |
| 0.081 | 1377.0 | 207.0 | 1752.0 | 0.004 |

Table 1.4 continues

| | | | | |
|-------|--------|-------|--------|-------|
| 0.026 | 1418.0 | 220.0 | 1102.0 | 0.002 |
| 0.041 | 1411.0 | 190.0 | 1215.0 | 0.002 |
| 0.035 | 1392.0 | 250.0 | 1121.0 | 0.001 |
| 0.029 | 1392.0 | 210.0 | 1122.0 | 0.004 |
| 0.065 | 1397.0 | 222.0 | 1731.0 | 0.003 |
| 0.070 | 1378.0 | 223.0 | 1789.0 | 0.001 |
| 0.065 | 1394.0 | 223.0 | 1738.0 | 0.004 |
| 0.025 | 1403.0 | 235.0 | 1149.0 | 0.003 |
| 0.032 | 1448.0 | 215.0 | 1213.0 | 0.002 |
| 0.066 | 1398.0 | 224.0 | 1757.0 | 0.005 |
| 0.075 | 1392.0 | 221.0 | 1819.0 | 0.004 |
| 0.080 | 1379.0 | 236.0 | 1651.0 | 0.003 |
| 0.019 | 1416.0 | 205.0 | 400.0 | 0.003 |
| 0.021 | 1444.0 | 215.0 | 937.0 | 0.001 |
| 0.024 | 1458.0 | 205.0 | 977.0 | 0.004 |

| Input number | Minimum Value | Maximum Value |
|---------------|---------------|---------------|
| 1 | 0.001 | 324.6 |
| 2 | 1570 | 1685 |
| 3 | 87 | 713 |
| 4 | 140 | 393 |
| 5 | 7 | 342 |
| 6 | 300 | 1678 |
| 7 | 12330 | 15250 |
| Output Number | Minimum Value | Maximum Value |
| 1 | 7 | 26 |

**Table 4.1 : Minimum and maximum values of scaling for
Dephosphorization data**

| Input number | Minimum Value | Maximum Value |
|---------------|---------------|---------------|
| 1 | 0.014 | 0.125 |
| 2 | 1343 | 1450 |
| 3 | 150 | 270 |
| 4 | 383 | 2000 |
| Output Number | Minimum Value | Maximum Value |
| 1 | 0.001 | 0.069 |

**Table 4.2 : Minimum and maximum values of scaling for
Desulphurization data**

| $\alpha = 0.5$; $\beta = 0.8$; $g = 0.7$ No of training examples = 200 No of testing examples = 96 | | | |
|--|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 7-6-1 | 0.076323 | 0.832894 | 0.539198 |
| 7-8-1 | 0.079208 | 0.801864 | 0.695834 |
| 7-10-1 | 0.078031 | 0.801864 | 0.580312 |
| 7-12-1 | 0.068739 | 0.841213 | 0.685473 |
| 7-6-6-1 | 0.054583 | 0.870372 | 0.603883 |
| 7-6-8-1 | 0.044602 | 0.864173 | 0.511987 |
| 7-8-8-1 | 0.040922 | 0.931688 | 0.249237 |

**Table 4.3 : Results of Backpropagation on
Dephosphorization Problem :**

| $\alpha = 0.5$; $\beta = 0.8$; $g = 0.7$ No of training examples = 200 No of testing examples = 96 | | | |
|--|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 7-6-1 | 0.079049 | 0.831192 | 0.806281 |
| 7-8-1 | 0.075946 | 0.831193 | 0.716036 |
| 7-10-1 | 0.073926 | 0.838790 | 0.644370 |
| 7-12-1 | 0.088838 | 0.772048 | 0.859380 |
| 7-6-6-1 | 0.051753 | 0.948962 | 0.488660 |
| 7-6-8-1 | 0.052508 | 0.940668 | 0.587063 |
| 7-8-8-1 | 0.042091 | 0.961218 | 0.552343 |

**Table 4.4 : Results of Backpropagation -Variation 1 on
Dephosphorization Problem :**

| $\alpha = 0.5$; $\beta = 0.8$; $g = 0.7$ No of training examples = 200 No of testing examples = 96 | | | |
|--|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 7-6-1 | 0.128147 | 0.622588 | 0.373135 |
| 7-8-1 | 0.124840 | 0.654092 | 0.585130 |
| 7-10-1 | 0.118120 | 0.656587 | 0.521535 |
| 7-12-1 | 0.117039 | 0.656587 | 0.519539 |
| 7-6-6-1 | 0.102579 | 0.647174 | 0.517539 |
| 7-6-8-1 | 0.092768 | 0.691199 | 0.649539 |
| 7-8-8-1 | 0.105020 | 0.540999 | 0.458165 |

**Table 4.5 : Results of Backpropagation -Variation 2 on
Dephosphorization Problem :**

| $\alpha = 0.5$; $\beta = 0.8$; $g = 0.7$ No of training examples = 200 No of testing examples = 96 | | | |
|--|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 7-6-1 | 0.082202 | 0.779982 | 0.422928 |
| 7-8-1 | 0.129565 | 0.745827 | 0.715184 |
| 7-10-1 | 0.133943 | 0.738052 | 0.725899 |
| 7-12-1 | 0.103374 | 0.792700 | 0.485438 |
| 7-6-6-1 | 0.103374 | 0.892290 | 0.464322 |
| 7-6-8-1 | 0.074621 | 0.916567 | 0.669145 |
| 7-8-8-1 | 0.102514 | 0.889374 | 0.649643 |

**Table 4.6 : Results of Backpropagation -Variation 3 on
Dephosphorization Problem :**

| $\alpha = 0.01$; $\beta = 0.01$; $c = 1.0$ No of training examples = 200 No of testing examples = 96 | | | |
|--|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 7-6-1 | 0.076134 | 0.793873 | 0.860321 |
| 7-8-1 | 0.076822 | 0.794515 | 0.853842 |
| 7-10-1 | 0.076588 | 0.794595 | 0.859187 |
| 7-12-1 | 0.074340 | 0.806136 | 0.860216 |
| 7-6-6-1 | 0.077704 | 0.788124 | 0.862793 |
| 7-6-8-1 | 0.073387 | 0.811953 | 0.850981 |
| 7-8-8-1 | 0.079524 | 0.800564 | 0.856852 |

**Table 4.7 : Results of Backpropagation -Variation 4 on
Dephosphorization Problem (on Scaled data):**

| $\alpha = 0.01$; $\beta = 0.01$; $c = 1.0$ No of training examples = 200 No of testing examples = 96 | | | |
|--|--|--|---|
| Constant c | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 1 | 0.137582 | 0.013615 | 0.000000 |
| 10 | 0.126219 | 0.014042 | 0.000000 |
| 50 | 0.126282 | 0.023850 | 0.000000 |
| 100 | 0.126223 | 0.078109 | 0.000000 |
| 500 | 0.126624 | 0.169631 | 0.225519 |
| 1000 | 0.126193 | 0.013959 | 0.000000 |

**Table 4.8 : Results of Backpropagation -Variation 4 on
Dephosphorization Problem (on Unscaled data):**

| $\alpha = 0.5$; $\beta = 0.8$; $g = 0.7$ No of training examples = 350 No of testing examples = 161 | | | |
|---|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 4-4-1 | 0.053349 | 0.310698 | 0.442279 |
| 4-6-1 | 0.053184 | 0.316880 | 0.409008 |
| 4-8-1 | 0.053304 | 0.312595 | 0.430579 |
| 4-10-1 | 0.053303 | 0.311930 | 0.422568 |
| 4-4-4-1 | 0.053531 | 0.296487 | 0.403366 |
| 4-4-6-1 | 0.053145 | 0.316859 | 0.419888 |
| 4-6-6-1 | 0.053148 | 0.308292 | 0.437557 |

**Table 4.9 : Results of Backpropagation on
Desulphurization Problem :**

| $\alpha = 0.5 ; \beta = 0.8 ; g = 0.7$ No of training examples = 350 No of testing examples = 161 | | | |
|---|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 4-4-1 | 0.064413 | 0.270909 | 0.433205 |
| 4-6-1 | 0.064303 | 0.182219 | 0.313586 |
| 4-8-1 | 0.063222 | 0.178613 | 0.239764 |
| 4-10-1 | 0.064538 | 0.227376 | 0.035725 |
| 4-4-4-1 | 0.060011 | 0.356236 | 0.311480 |
| 4-4-6-1 | 0.061817 | 0.249529 | 0.171578 |
| 4-6-6-1 | 0.060801 | 0.284168 | 0.003358 |

**Table 4.11 : Results of Backpropagation -Variation 2 on
Desulphurization Problem :**

| $\alpha = 0.5$; $\beta = 0.8$; $g = 0.7$ No of training examples = 350 No of testing examples = 161 | | | |
|---|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 4-4-1 | 0.070117 | 0.257120 | 0.170883 |
| 4-6-1 | 0.096675 | 0.228698 | 0.000000 |
| 4-8-1 | 0.070175 | 0.353635 | 0.000000 |
| 4-10-1 | 0.081772 | 0.288816 | 0.025347 |
| 4-4-4-1 | 0.058078 | 0.130531 | 0.104026 |
| 4-4-6-1 | 0.047505 | 0.644722 | 0.064219 |
| 4-6-6-1 | 0.081218 | 0.000000 | 0.000000 |

**Table 4.12 : Results of Backpropagation -Variation 3 on
Desulphurization Problem :**

| $\alpha = 0.5$; $\beta = 0.8$; $c = 1.0$ No of training examples = 350 No of testing examples = 161 | | | |
|---|--|--|---|
| Network Config- eration | Learning Error after 10000 Iteration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 4-4-1 | 0.053237 | 0.279083 | 0.512769 |
| 4-6-1 | 0.053514 | 0.276960 | 0.510887 |
| 4-8-1 | 0.053435 | 0.282473 | 0.496166 |
| 4-10-1 | 0.053496 | 0.279486 | 0.520799 |
| 4-4-4-1 | 0.053472 | 0.280845 | 0.510440 |
| 4-4-6-1 | 0.053475 | 0.279849 | 0.510440 |
| 4-6-6-1 | 0.053513 | 0.276460 | 0.512586 |

**Table 4.13 : Results of Backpropagation -Variation 4 on
Desulphurization Problem (On scaled Data)**

| No of Neurons in middle layer = no of centers | | | |
|---|-------------------------|--|---|
| Network Config- eration | Final Learning Error | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 7-5-1 | 0.108388 | 0.512403 | 0.603914 |
| 7-10-1 | 0.092750 | 0.678223 | 0.736951 |
| 7-15-1 | 0.091228 | 0.691067 | 0.742099 |
| 7-20-1 | 0.087777 | 0.718576 | 0.817696 |
| 7-25-1 | 0.086634 | 0.727229 | 0.796878 |
| 7-30-1 | 0.083707 | 0.748436 | 0.755894 |
| 7-35-1 | 0.079480 | 0.776827 | 0.742612 |
| 7-50-1 | 0.073253 | 0.814345 | 0.774100 |
| 7-60-1 | 0.072183 | 0.820320 | 0.785239 |
| 7-70-1 | 0.068020 | 0.820302 | 0.785239 |
| 7-80-1 | 0.064911 | 0.857621 | 0.738984 |
| 7-100-1 | 0.058396 | 0.886532 | 0.717928 |
| 7-199-1 | 0.000000 | 1.000000 | 0.665212 |

Table 4.14 : Results of RBF on Dephosphorization Problem

| No of Neurons in middle layer = no of centers | | | |
|---|-------------------------|--|---|
| Network Config- eration | Final Learning Error | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| 4-10-1 | 0.053798 | 0.254834 | 0.359159 |
| 4-20-1 | 0.053492 | 0.274844 | 0.413109 |
| 4-30-1 | 0.052438 | 0.334108 | 0.414418 |
| 4-33-1 | 0.051897 | 0.360351 | 0.350241 |
| 4-35-1 | 0.051825 | 0.363675 | 0.332051 |
| 4-40-1 | 0.051546 | 0.376280 | 0.315539 |
| 4-50-1 | 0.050034 | 0.437265 | 0.251655 |
| 4-100-1 | 0.043975 | 0.612558 | 0.251655 |
| 4-349-1 | 0.000000 | 1.000000 | 0.040887 |

Table 4.15 : Results of RBF on Desulphurization Problem


```

Population Size =1000
No of Generation = 500
Crossover probability = 0.9
Mutation probability  = 0.085
Crossover Used = Multiple point crossover with
                  no of sites = 4
Selection Scheme = Tournament Selection (binary )
No of Backpropagation runs after GA = 1000
learning rate = 0.8
Momentum factor = 0.5

```

| | | | | |
|--|----------|----------|----------|-----------|
| Network Configur- ation | 4-4-1 | 4-6-1 | 4-8-1 | 4 - 10 -1 |
| Best ever found in generation | 267 | 379 | 229 | 67 |
| Learning Error after running GA | 0.054242 | 0.056878 | 0.056005 | 0.057884 |
| Learning Error after 1000 iter. of BP | 0.053802 | 0.056953 | 0.055538 | 0.057339 |
| Training Data Correlation Coefficient | 0.263367 | 0.116777 | 0.138314 | 0.116329 |
| Testing data Correlation Coefficient | 0.447047 | 0.377225 | 0.216346 | 0.200927 |

Table 4.17 : Results of GA+ BP on desulphurization problem

| | | | | |
|--|----------|----------|----------|--|
| Population Size =1000 No of Generation = 500 Crossover probability = 0.9 Mutation probability = 0.085 Crossover Used = Multiple point crossover with no of sites = 4 Selection Scheme = Tournament Selection (binary) No of Backpropagation runs after GA = 1000 learning rate = 0.8 Momentum factor = 0.5 | | | | |
| Network Configur- ation | 4-4-4-1 | 4-4-6-1 | 4-6-6-1 | |
| Best ever found in generation | 108 | 184 | 325 | |
| Learning Error after running GA | 0.055730 | 0.055597 | 0.055635 | |
| Learning Error after 1000 epochs of BP | 0.055086 | 0.055160 | 0.054913 | |
| Training Data Correlation Coefficient | 0.225060 | 0.205680 | 0.215004 | |
| Testing data Correlation Coefficient | 0.213897 | 0.257842 | 0.316852 | |

Table 4.17 continued :

Results of GA+ BP on desulphurization problem

| Results of Network of type Fig 3.4 (a) (ANFIS) Number of Membership function (MFs) associated = r Step size =0.5 | | | |
|--|--|--|---|
| Network Configuration (no of Membership associated with each input) | Final Learning Error after 1000 Iteration | Training Data Performance Correlation Coefficient | Testing data Performance Correlation Coefficient |
| 7-14-2-2-2-1 (2 MFs) | 0.071093 | 0.826277 | 0.752207 |
| 7-21-3-3-3-1 (3 MFs) | 0.069264 | 0.835972 | 0.414561 |
| 7-28-4-4-4-1 (4 MFs) | 0.064561 | 0.859272 | 0.000000 |
| 7-35-5-5-5-1 (5 MFs) | 0.062912 | 0.866921 | 0.000000 |
| 7-42-6-6-6-1 (6 MFs) | 0.060010 | 0.879742 | 0.000000 |
| 7-56-8-8-8-1 (8 MFs) | 0.042474 | 0.941677 | 0.223721 |
| 7-70-10-10-10-1 (10 MFs) | 0.036695 | 0.956409 | 0.000000 |
| 7-105-15-15-15-1 (15 MFs) | 0.038900 | 0.951322 | 0.0452184 |

Table 4.18 : Results of ANFIS (Network of type Fig (3.4 a))
on dephosphorization

| Results of Network of type Fig (3.2 a) (ANFIS) Number of Membership function (MFs) associated = r Step size =0.5 | | | |
|--|--|--|---|
| Network Configuration (no of MFs associated with each input) | Final Learning Error after 1000 Iteration | Training Data Performance Correlation Coefficient | Testing data Performance Correlation Coefficient |
| 4-8-2-2-2-1 (2 MFs) | 0.052296 | 0.341197 | 0.563400 |
| 4-16-4-4-4-1 (4 MFs) | 0.046978 | 0.535738 | 0.304037 |
| 4-24-6-6-6-1 (6 MFs) | 0.049276 | 0.464283 | 0.000000 |
| 4-32-8-8-8-1 (8 MFs) | 0.033096 | 0.803817 | 0.000000 |
| 4-48-12-12-12-1 (12 MFs) | 0.043923 | 0.613778 | 0.006811 |
| 4-64-16-16-16-1 (16 MFs) | 0.018306 | 0.944312 | 0.000000 |
| 4-80-20-20-20-1 (20 MFs) | 0.014663 | 0.964641 | 0.042546 |
| 4-120-30-30-30-1 (30 MFs) | 0.011865 | 0.976994 | 0.0416629 |

Table 4.19 :Results of ANFIS (Network of type Fig (3.4 a))
on desulphurization problem

| Results of Network of type Fig (3.2 b) (ANFIS) Number of Membership function (MFs) associated = r Step size =0.5 | | | |
|--|--|--|---|
| Network Configuration (no of MFs associated with each input) | Final Learning Error after 1000 Iteration | Training Data Performance Correlation Coefficient | Testing data Performance Correlation Coefficient |
| 4-8-16-16-16-1 (2 MFs) | 0.048607 | 0.486509 | 0.378011 |

**Table 4.20 :Results of ANFIS (Network of type Fig (3.4 b))
on desulphurization problem**

| | | | |
|---|-----------------------|-----------------------|-----------------------|
| Population Size = 50 No of Generation = 100 Crossover probability = 0.9 Mutation probability = 0.050 Crossover Used = Multiple point crossover with no of sites = 4 Selection Scheme = Tournament Selection (binary) Number of Membership function (MFs) associated = r | | | |
| Number of MF associated/ Network Configura- tion | 2 MFs 7-14-2-2-2-1 | 3 MFS 7-21-3-3-3-1 | 4 MFs 7-28-4-4-4-1 |
| Best ever found in generation | 63 | 48 | 45 |
| Learning Error form best ever solution | 0.065554 | 0.061689 | 0.058639 |
| Training Data Correlation Coefficient | 0.854943 | 0.872422 | 0.885526 |
| Testing data Correlation Coefficient | 0.860947 | 0.800141 | 0.794104 |

**Table 4.21 : Results of GA assisted ANFIS
(Network of type Fig (3.4 a))
on dephosphorization**

| | | | |
|---|-----------------------|-----------------------|-----------------------|
| Population Size = 50 No of Generation = 100 Crossover probability = 0.9 Mutation probability = 0.050 Crossover Used = Multiple point crossover with no of sites = 4 Selection Scheme = Tournament Selection (binary) Number of Membership function (MFs) associated = r | | | |
| Number of MF associated/ Network Configuration | 5 MFs 7-35-5-5-5-1 | 6 MFs 7-42-6-6-6-1 | 8 MFs 7-56-8-8-8-1 |
| Best ever found in generation | 51 | 52 | 49 |
| Learning Error form best ever solution | 0.055466 | 0.052028 | 0.045657 |
| Training Data Correlation Coefficient | 0.898267 | 0.911087 | 0.932282 |
| Testing data Correlation Coefficient | 0.819520 | 0.357725 | 0.285631 |

**Table 4.21 continued : Results of GA assisted ANFIS
(Network of type Fig (3.4 a))**

on dephosphorization

| | | | |
|--|----------------------|-----------------------|-----------------------|
| Population Size =50 No of Generation = 100 Crossover probability = 0.9 Mutation probability = 0.050 Crossover Used = Multiple point crossover with no of sites = 4 Selection Scheme = Tournament Selection (binary) Results of GA assisted ANFIS of type Fig (3.4 a) Number of Membership function (MFs) associated = r | | | |
| Number of MF associated /Network Configura- tion | 2 MFs 4-8-2-2-2-1 | 4 MFS 4-16-4-4-4-1 | 6 MFS 4-24-6-6-6-1 |
| Best ever found in generation | 22 | 74 | 78 |
| Learning Error form best ever solution | 0.035346 | 0.034239 | 0.034469 |
| Training Data Correlation Coefficient | 0.772259 | 0.788209 | 0.784964 |
| Testing data Correlation Coefficient | 0.108226 | 0.342433 | 0.000000 |

**Table 4.22 : Results of GA assisted ANFIS
(Network of type Fig (3.4 a))
on desulphurization problem**

| | | |
|--|-------------------------------|--------------------------------|
| Population Size =50 No of Generation = 100 Crossover probability = 0.9 Mutation probability = 0.050 Crossover Used = Multiple point crossover with no of sites = 4 Selection Scheme = Tournament Selection (binary) Results of GA assisted ANFIS of type Fig (3.4 a) Number of Membership function (MFs) associated = r | | |
| Number of MF associated /Network Configura- tion | 20 MFs 4-80-20-20-20-1 | 30 MFs 4-120-30-30-30-1 |
| Best ever found in generation | 98 | 77 |
| Learning Error form best ever solution | 0.028738 | 0.023738 |
| Training Data Correlation Coefficient | 0.856266 | 0.905593 |
| Testing data Correlation Coefficient | 0.227856 | 0.074318 |

**Table 4.22 continued : Results of GA assisted ANFIS
(Network of type Fig (3.4 a))
on desulphurization problem**

| Comparison of Best performance of Various Algorithms | | | |
|--|-----------------------|---|--|
| Algorithm | Network Configuration | Training Data Performance Correlation Coefficient | Testing Data Performance Correlation Coefficient |
| BP | 7-16-1 | 0.841 | 0.685 |
| Variation of BP in Var 1 | 7-6-1 | 0.820 | 0.785 |
| RBF | 7-60-1 | 0.820 | 0.785 |
| GA + BP | 7-8-8-1 | 0.737 | 0.853 |
| ANFIS | 2 MF 7-14-2-2-2-1 | 0.826 | 0.752 |
| GA+ ANFIS | 2 MF 7-14-2-2-2-1 | 0.854 | 0.861 |

**Table 4.24 : Comparison of all algorithms on
Dephosphorization Problem :**

